

## On the Evaluation of Agent Behaviors\*

Amol Dattatraya Mali

Dept. of Electrical Engg. & Computer Science,

University of Wisconsin, Milwaukee, WI 53211, USA

Phone: 1-414-229-6762, Fax: 1-414-229-2769, mali@millers.cs.uwm.edu

This work focuses on robot behaviors which use minimal communication and rely mostly on changes in the environment as their cue for action. The behavior-based paradigm for building autonomous robots has become very popular because of its successes, use of the world as an external memory and replacement of classical planning by agent-environment dynamics. However there are no criteria for evaluating and improving behavior sets. Our aim here is to bridge this gap. We define several criteria (power, usefulness, flexibility, modularity, and, reliability) and investigate the properties of behavior sets using them. We use these criteria to present results on modifications to individual behaviors and addition of new behaviors to the behavior sets. We show how computations related to these criteria can be carried out. We report on guidelines to improve a behavior set.

**Keywords** - Autonomy, Evaluation, Agency, Robotics, Behaviors

### 1. Introduction

Classical planners synthesize a sequence of actions that can be executed from a given initial state of the world to achieve a given goal. Classical planners explore a combinatorially large search space. Thus, not surprisingly, classical planning has been proved to be intractable [6]. Behavior-based 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 behavior-based robot is a collection of several independent task-achieving modules (behaviors) with a simple distributed control mechanism. Each behavior mediates directly with the external world. The behaviors are 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. All behavior-based systems attempt to reduce or eliminate the centralized shared memory, relying instead on parameter passing and communication between individual behaviors.

Interesting results have been achieved using behavior-based robots in the can collection task [7], navigation of a mobile robot [1], and, a 6 legged walking robot [13]. The autonomous rover *Rockey-III* that navigates through rough outdoor terrain and collects soil

---

\*Appears in *Artificial Intelligence Journal*, Vol. 143, No. 1, Jan. 2003, pp. 1-17

samples [16] is behavior-based. The flying vehicle that won the aerial robotics competition [17] is behavior-based. The 12 degree of freedom hyper-redundant serpentine robot that navigates complex pipe structures and inspects them [8] is behavior-based. The walking robot for exploring volcanic craters [20], and the drum sampling robot [12] are behavior-based. The mobile manipulator [5] and the non-holonomic robot for box pushing and ball dribbling [9] are also behavior-based. The mobile security guards [3], SONY's robotic dog for entertainment [2], and the robot that learns to push boxes [14] are behavior-based as well. Additional work on behavior-based robots includes [10].

The behavior-based agency had a large impact on artificial intelligence. Several journal issues have 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 behavior-based agency, traditional AI researchers have argued that behavior-based control cannot serve as a complete substitute for a global representation [11]. Others have attacked the parallel psychological model of situated action, showing that the behavior-based models were actually symbol systems. In other words, the models required significant internal representations which were, or could be represented using, symbolic systems [19]. We find missing in this debate a theory of purely behavior-based agency that takes into account its potential. There are no well-defined criteria for comparing different behavior systems. The design of behaviors is an ad-hoc process. Different designers come up with different sets of behaviors for the same task. How should these behavior sets be compared and evaluated? How should they be modified to achieve a different functionality? How should the effects of such modifications be evaluated? This is not clear. In the current implementations, if the behaviors do not work, either the environment is modified or some task-specific or environment-specific heuristics are added to the programs. Though the testing of the behaviors in the real world is essential, it currently lacks formal foundations. We set out to answer these questions and to help bridge the gap between the design of behavior sets and their operation in the real world.

This work makes the following contributions:

- We define five criteria to evaluate a behavior set. They are: power, usefulness, modularity, flexibility, and, reliability.
- We introduce the notion of a greatest potential task space. This allows a user to know the task-fulfilling capability and to modify a behavior set.
- We report on guidelines for modifying a behavior set to improve either its usefulness, flexibility, modularity, or reliability.

This paper is organized as follows: In section 2, we define several criteria for comparing behaviors, behavior chains, and behavior sets. In section 3, we examine the properties of the behaviors, the behavior chains, and the sets of behaviors using the criteria defined in section 2. In section 4, we investigate the effect of some modifications applied to the stimuli and consequences of behaviors, on the properties of the behaviors, on behavior chains, and on the behavior sets. In section 5, we show how computations related to various evaluation criteria can be carried out. We also report on several guidelines for improving behavior sets. In section 6, we discuss a behavior-based kitchen management robot to illustrate the practical importance of our results. Some comments on our behavior evaluation criteria and on directions for future work are discussed in section 7. We present

our conclusions in section 8. Proofs of our results and additional results appear in [15].

## 2. Behaviors & Evaluation Criteria

We consider a system of agents in this work. All changes to the world are caused by an action of one of these agents, which are interchangeably called behaviors. Nothing changes the world except the agents. The essential notion of a behavior is a mapping from a stimulus to a consequence. We use the *3-tuple* model of behavior: stimulus, action, consequence. Thus, a behavior  $\beta_i$  is denoted by  $\langle s_i, a_i, c_i \rangle$ , where  $s_i$  denotes stimulus of  $\beta_i$ ,  $a_i$  denotes the action of  $\beta_i$  and  $c_i$  denotes the consequence of  $\beta_i$ . In the examples and results that follow,  $s_i$  and  $c_i$  are defined using first-order predicates as in Brooks' subsumption architecture. In this paper, the stimuli and consequences are assumed to be in a purely conjunctive form. We assume that, for successful execution of a behavior, all predicates in its stimulus should be true at the start of its execution. We also assume that the predicates in the consequence of a behavior are true at the end of its execution. Furthermore, the truths of the predicates in the consequence of a behavior are undefined during its execution. And, the consequences of behaviors are certain. For example, after the execution of the behavior of picking up a block, the block is assumed to be in the robot's gripper.

### 2.1. Behavior Chain

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

- **Behavior Chain:** a temporal sequence of behavior modules  $\{\beta_{i_1} : \beta_{i_2} : \beta_{i_3} : \dots : \beta_{i_n}\}$ . The actions of earlier modules change the situation in such a way that the newly changed part of the situation in conjunction with the unchanged part provides stimulus for the next module in the sequence. Here “:” denotes temporal contiguity and  $\prec$  denotes a temporal precedence relation.

- **Task:** A task is defined as a transition from one state of the world to another state, achieved through a temporal chain of behaviors. A task is specified as  $\langle I, G \rangle$  where  $I$  is the conjunction of predicates true in the state of world from which a state containing the conjunction of predicates  $G$  needs to be achieved. Predicates not specified in  $I$  or  $G$  may be either true or false. Some of these predicates may be irrelevant to the task.  $I$  is the initial state of the task and  $G$  is its final state.

For two tasks to differ, their initial and/or final states must differ. Thus, the chain  $\{\textit{open\_water\_tap} : \textit{close\_water\_tap}\}$  fulfills the same task as the chain  $\{\textit{open\_water\_tap} : \textit{close\_water\_tap} : \textit{open\_water\_tap} : \textit{close\_water\_tap}\}$ .

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 that lead to different final states and/or execute from different initial states. A **task space** is the space of tasks. When tasks are completed by executing behaviors in a temporal sequence, the size of the task space can be found by counting the number of executable chains that complete different state transitions. Thus, the chain  $\{\beta_1 : \beta_8 : \beta_9\}$  may represent a task that is different from the task represented by  $\{\beta_1 : \beta_8\}$ . A behavior chain that continues to fulfill a specific task even after the removal of some behaviors is called

a non-minimal chain.

We use the following symbols from first-order predicate calculus:  $\exists$  (existential quantifier),  $\forall$  (universal quantifier),  $\wedge$  (logical AND),  $\neg$  (logical negation),  $\Rightarrow$  (logical implication), and  $\vee$  (logical OR). A literal is an atomic formula or the negation of an atomic formula. A predicate is an atomic formula. A ground, or fully instantiated, predicate is a predicate with all its variables replaced by constants. A ground predicate evaluates to true or false. For example,  $\text{on}(x,y)$  is a predicate that represents object  $x$  being on object  $y$ .  $\text{on}(A,B)$ , where  $A$  and  $B$  are specific blocks, is a ground predicate. A clause is a disjunction of literals. For example,  $(\text{glass}(A) \vee \text{glass}(B) \vee \text{glass}(C))$  is a clause.

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 predicates will be affected by the actions in a behavior chain; the rest constitute the *universe*, which, throughout 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 the predicates from  $U$ . We limit ourselves to those predicates whose change affects the firing of other behaviors in the chain.

What we mean by a “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  $\beta_1$  starting in  $(X \wedge Y \wedge Z)$  makes  $A$  true.  $c_1$ ,  $\beta_1$ ’s consequence, is assumed to contain  $X$ , a part of the universe. Then  $\beta_1$  leads to  $\beta_2$  since  $(c_1 \Rightarrow s_2)$ . Thus, when we say that  $\{\beta_i : \beta_j\}$ , we mean that a part of  $s_j$  was true in the Universe and that the execution of  $\beta_i$  caused the rest of  $s_j$  to become true. This is another version of the frame problem, which arises in any model involving temporal sequencing, such as the add and delete list model used widely in planning and knowledge representation. The model chosen here can be reformulated in terms of the successor state axioms that Reiter has shown to be derivable from the positive and negative effect axiomatic structure [18].

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 the elements of  $B$ .  $|C|$  denotes length of the chain  $C$  (which is same as the number of behavior modules in the chain). The members of a behavior set are denoted by  $\beta_1, \beta_2, \dots, \beta_{|B|}$ .  $\beta_i \in C$  denotes that  $\beta_i$  occurs in the chain  $C$ .

Concurrent behaviors that do not interfere can be handled in the sequential chain model by considering the cases in Fig. 1. We adopted the chaining model to analyze autonomous behavior since the primary paradigm for solving most AI problems is to construct a sequence of actions that can be executed from initial state to reach the goal. The achievement of a task by a behavior-based robot can always be shown to be due to the formation of a behavior sequence or to the execution of parallel behavior sequences.

We show next how various temporal overlaps between concurrent behaviors can be incorporated in the behavior chain model for the purpose of analysis. In the process, we will discuss seven different temporal relationships between behaviors (shown in Figure 1). These temporal relations are useful for analysis when they are known a priori. In that case our treatment below can be used to include concurrent behaviors in the chaining model. Though exact temporal relations between behaviors will not be known a priori in practice,

the nature of such relationships can be predicted based on the consequences and stimuli of behaviors. And, the behavior chain model can still be used to detect potential undesired behavior (e.g. infinite cycles) and to find possibilities for improvement of behaviors or chains.

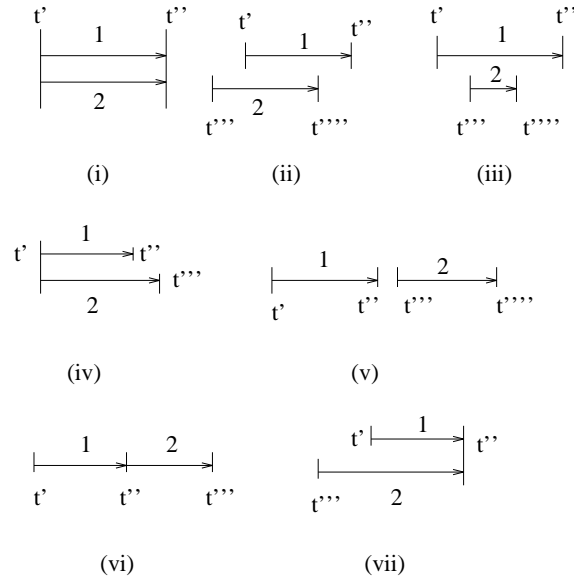


Figure 1. Temporal Relations between behaviors  $\beta_1$  and  $\beta_2$ . Temporal spans of these are denoted by 1, 2 respectively.

Temporal relations between  $\beta_1$  and  $\beta_2$  may differ due to the precedence relations between the start and end points of the executions of the two behaviors. We refer to the difference between the start and end time of a behavior's execution as its "temporal span". The temporal relations from Fig. 1 are discussed next.

(i) The consequences  $c_1$  and  $c_2$  of both the behaviors will be true at the end of their execution (at time  $t''$ ) if the following three conditions hold: (a) The consequence of  $\beta_1$  does not contradict the consequence of  $\beta_2$ , that is,  $c_1$  does not contain a predicate or a negated predicate that is false in  $c_2$ , (b)  $\beta_1$  does not make any predicate in the stimulus of  $\beta_2$  false, and, (c)  $\beta_2$  does not make any predicate in the stimulus of  $\beta_1$  false. This concurrency can be treated as either  $\beta_1 : \beta_2$  or  $\beta_2 : \beta_1$  and included in the chain. This case of concurrent execution can also be represented by a virtual behavior  $\beta''$  such that its stimulus and consequence are  $s'' = (s_1 \wedge s_2)$  and  $c'' = (c_1 \wedge c_2)$  respectively.  $\beta''$  can then be included in a chain. The temporal span of  $\beta''$  is  $t'' - t'$ . Obviously, the virtual behavior will have to be replaced by  $\beta_1$  and  $\beta_2$  at the time of execution,

(ii)  $s_1$  is true at  $t'$ .  $s_2$  is true at  $t'''$ . The consequence  $c_1$  of  $\beta_1$  will be true at  $t''$ .  $c_2$  will be true at  $t''''$ . For all this to hold, it is necessary that (a)  $\beta_1$  does not falsify any predicate in  $s_2$  or  $c_2$ , and that (b)  $\beta_2$  does not falsify any predicate in  $s_1$  or  $c_1$ .

(iii), (iv), (vii) These can be handled in the same way as (ii).

(v) This case is captured in the definition of a behavior chain. The temporal gap  $t''' - t''$  is a period during which either no behavior is executed or the default behavior of wandering occurs. Case (vi) is a special case of (v) where there is no temporal gap.

## 2.2. Behavior Evaluation Criteria

To compare different behavior sets, we define several criteria that relate to the effectiveness of a behavior set.

- **Power**: A behavior  $\beta := \langle s, a, c \rangle$  is at least as powerful as another behavior  $\beta' := \langle s', a', c' \rangle$  iff  $(s' \Rightarrow s) \wedge (c \Rightarrow c')$ . In other words, if a behavior  $\beta$  can be triggered at least as frequently as another behavior  $\beta'$  and results in a consequence that subsumes  $\beta'$ 's consequence (Figure 2), then  $\beta$  is at least as powerful as  $\beta'$ . A behavior  $\beta$  is more powerful than another behavior  $\beta'$  if (i)  $\beta$  is at least as powerful as  $\beta'$ , and if (ii)  $\neg(s \Rightarrow s')$  or  $\neg(c' \Rightarrow c)$  is true.

A behavior set  $B$  is more powerful than another behavior set  $B'$  iff  $B'$  can be obtained from  $B$  by replacing some behavior  $\beta \in B$  by a less powerful behavior  $\beta'$ .  $B >_p B'$  denotes that the behavior set  $B$  is more powerful than the behavior set  $B'$ . The powers of three behaviors are compared in Fig. 2. The stimuli of these behaviors are also shown in Fig. 2. Though not shown in the figure, the stimulus of each of these behaviors also contains *gripper\_free*, since objects can be picked up only if a robot's gripper is free.  $x$  in the description of the stimuli is a variable. Behavior  $\beta_1$  can be executed if (i) there is an object such that it is graspable and is a can and (ii) the gripper of the robot is free. The consequence of these three behaviors is the same. It is  $(has\_gripper(x) \wedge \neg gripper\_free)$ .




Stimuli for behaviors of increasing power		
$(\beta_1)$	$(\beta_2)$	$(\beta_3)$
		
$graspable(x) \wedge can(x)$	$graspable(x) \wedge (can(x) \vee cup(x))$	$graspable(x)$

Figure 2. *Power*: Behavior  $\beta_1$  can pick up only cans. Behavior  $\beta_2$  can pick up cups as well as cans (has a more general stimulus).  $\beta_2$  is more powerful than  $\beta_1$ .  $\beta_3$  is even more powerful. The notion of power is used to capture this. All three behaviors have the same consequence.

- **Greatest Potential Task Space**  $\tau_G(B)$ : The behavior set  $B$  **spans** the task space  $\tau$  if  $\forall(t \in \tau) (\exists(C \triangleleft B) fulfills(C, t))$ , where  $t$  is a task. This means that if  $B$  spans  $\tau$ , then for each task in  $\tau$ , there is a chain composable from behaviors in  $B$  such that the chain fulfills the task.  $\tau_G(B)$  is the set of all tasks that can be fulfilled by behaviors in  $B$ , i.e. the largest task space that is spanned by  $B$ .

- **Usefulness**: The ratio  $\frac{|\tau_G(B)|}{|B|}$ .  $B >_u B'$  denotes that the behavior set  $B$  is more

useful than the behavior set  $B'$ . The usefulness of the behavior set  $B_i$  is denoted by  $U(B_i)$ .

- **Flexibility:** A behavior set  $B$  is at least as flexible as a behavior set  $B'$  iff  $\forall t \in (\tau_G(B) \cap \tau_G(B')) (\exists (C \triangleleft B) (fulfills(C, t) \wedge \forall (C' \triangleleft B') (fulfills(C', t) \Rightarrow |C| \leq |C'|)))$ . This means that  $B$  can fulfill tasks with as short or shorter chains than  $B'$  can.  $B \geq_f B'$  denotes that the behavior set  $B$  is at least as flexible as the behavior set  $B'$ . Flexibility is important since it is directly related to task completion times. Longer behavior chains may need more time to execute than shorter behavior chains. Power and Flexibility are relative criteria.

- **Modularity:** A behavior set is more modular if different modules in that set are more independent, i.e. there is minimal interference between them. One measure of interference in a behavior set is the incidence of cyclic behavior (cyclic conflict is defined in section 3). We therefore define the modularity of a behavior set  $B$  as the inverse of the likelihood that cyclic conflicts will arise in  $B$ . We define the likelihood of a cycle in  $B$ , denoted by  $L_{cycle}(B)$ , as

$$\sum_{C \in \tau_G(B)} \sum_{(\beta_i \prec \beta_j, \beta_i \in C, \beta_j \in C)} (prob(c_j \Rightarrow s_i))$$

Thus, the likelihood is found by considering all pairs of behaviors in all chains in  $\tau_G(B)$ .  $\beta_i \prec \beta_j$  denotes that the behavior  $\beta_i$  precedes the behavior  $\beta_j$  in the chain.  $prob(c_j \Rightarrow s_i)$  denotes the probability that  $c_j$  implies  $s_i$ . It's 1 if  $c_j \Rightarrow s_i$  is true and 0 otherwise. However, the likelihood is not restricted to the 0-1 range like probabilities are.  $B \geq_m B'$  denotes that the behavior set  $B$  is at least as modular as the behavior set  $B'$ .

- **Reliability:** The probability of forming a behavior chain of length  $k$  using behaviors from  $B$  is greater than or equal to  $\frac{1}{|B|^k}$ . This is because there are upto  $|B|^k$  behavior chains of length  $k$  that are composable from behaviors in  $B$ . Reliability is a relative criterion. The reliability of a behavior set  $B$  is defined using the probability of composing chains that will fulfill the given tasks. Since there is no planning module in a purely behavior-based system, a task is fulfilled only if a chain of behaviors which achieves the desired state transformation is formed. The probability of forming such a chain is higher in a more reliable behavior set.  $prob(C)$  denotes the probability of forming a chain  $C$ . Behavior set  $B$  is at least as reliable as the behavior set  $B'$  if  $\forall t \in (\tau_G(B) \cap \tau_G(B')) \exists C \triangleleft B, C' \triangleleft B' (fulfills(C, t) \wedge fulfills(C', t) \wedge (prob(C) \geq prob(C')))$ .  $B$  is more reliable than  $B'$  if (i)  $B$  is at least as reliable as  $B'$  and (ii)  $\exists t \in (\tau_G(B) \cap \tau_G(B')) \exists C \triangleleft B, C' \triangleleft B' (fulfills(C, t) \wedge fulfills(C', t) \wedge (prob(C) > prob(C')))$ .

- **Scalability:** We define two disjoint behavior sets  $B$  and  $B'$  to be *scalable* together if  $\tau_G(B \cup B') \supset (\tau_G(B) \cup \tau_G(B'))$ .

In section 5 we will show how the computations needed to apply these criteria can be carried out.

### 3. Properties of Behavior Sets

In this section, we report results on relations between behavior sets, followed by definitions of various behavior conflicts.

**Lemma 1** If  $B_1 >_p B_2$  and  $B_2 >_p B_3$ , then  $B_1 >_p B_3$ .

**Lemma 2** If  $B_1 \geq_f B_2$ ,  $B_2 \geq_f B_3$ , and  $(\tau_G(B_1) \cap \tau_G(B_3)) \subseteq (\tau_G(B_1) \cap \tau_G(B_2))$ , then  $B_1 \geq_f B_3$ .

Next we will define various types of conflicts that may occur in behavior sets. In the broad sense of the word conflict, any behavior chain leading to non-fulfillment of desired objectives can be said to contain a conflict. Let the chain  $C = \{\beta_{i_1} : \beta_{i_2} : \dots : \beta_{i_n}\}$ ,  $1 \leq i_1, \dots, i_n \leq |B|$ ,  $C \triangleleft B$ , be a behavior sequence that achieves a desirable outcome. There are four types of conflicts that can cause the chain  $C$  to not be executed. In each case, some sequence  $\beta_{i_k} : \beta_{i_{k+1}}$ ,  $1 \leq k \leq (n - 1)$  is broken.

(a) **Extraneous behavior Conflict:**  $\beta_{i_k} : \beta'$ ,  $\beta' \notin C$ . This can be resolved by prioritization.

(b) **Cyclic Conflict:** This occurs due to an undesired repetition of a behavior chain, e.g.  $\{\beta_{i_p} : \beta_{i_{p+1}} : \dots : \beta_{i_{q-1}} : \beta_{i_q} : \beta_{i_p} : \beta_{i_{p+1}} : \dots : \beta_{i_q} : \beta_{i_p} : \dots\}$ . Here the chain preceding  $\beta_{i_q}$  executes again after  $\beta_{i_q}$  and the sequence  $\beta_{i_q} : \beta_{i_{q+1}}$  is broken.

(c) **Skipping Conflict:**  $\beta_{i_k} : \beta_{i_j}$ ,  $\beta_{i_j} \in C$ ,  $j > (k+1)$ . This type of conflict can be resolved by prioritization.

(d) **Control Conflict:** Stimuli of the behaviors  $\beta_{i_k}, \beta_{i_j}$ , both belonging to the same chain, are true simultaneously and these behaviors both need some same resource to execute. This conflict may lead to a deadlock if there is no arbitration mechanism. This conflict can be resolved by prioritization.

#### 4. Effects of Behavior Modifications

In this section, we investigate the effects of modifying the behaviors. Many times, stimuli are specialized to restrict the situations under which a behavior is triggered. For example, if the stimulus  $s_i$  of a behavior  $\beta_i$  for cleaning dishes is  $\exists x(dish(x) \wedge graspable(x) \wedge gripper\_free \wedge dirty(x))$  and it is desired that only those dishes that are on the table should be cleaned, then  $s_i$  can be specialized to  $\exists x, y(dish(x) \wedge dirty(x) \wedge graspable(x) \wedge gripper\_free \wedge table(y) \wedge on(x, y))$ . Also, the action of a behavior can be modified so that it results in a weaker consequence. We call this modification *consequence generalization*. For example, the consequence  $c_j$  of a behavior  $\beta_j$  for painting the side and top of a can is  $(painted(y) \wedge painted(z) \wedge \neg not\_painted(y) \wedge \neg not\_painted(z))$ .  $\beta_j$ 's stimulus  $s_j$  is  $\exists x, y, z (can(x) \wedge graspable(x) \wedge side(x, y) \wedge top(x, z) \wedge not\_painted(y) \wedge not\_painted(z))$ . The action of painting can be modified to paint only the top, resulting in the weaker consequence  $(painted(z) \wedge \neg not\_painted(z))$ . Stimulus specialization and consequence generalization can be used to eliminate undesirable cyclic conflicts. For example, consider the cycle  $\{\beta_{i_p} : \beta_{i_{p+1}} : \dots : \beta_{i_{q-1}} : \beta_{i_q} : \beta_{i_p} : \dots : \beta_{i_q} : \beta_{i_p} : \dots\}$ . This can be eliminated either by specializing  $s_{i_p}$  to  $s'_{i_p}$  such that  $(c_{i_q} \Rightarrow s'_{i_p})$  does not hold or by generalizing the consequence  $c_{i_q}$  to  $c'_{i_q}$  so that  $(c'_{i_q} \Rightarrow s_{i_p})$  does not hold. Both stimulus specialization and consequence generalization make a behavior less powerful. This leads to the next two theorems on behavior modification and modularity, respectively.

**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 of  $B$  by the less powerful ones) we have:

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

(b) The usefulness of  $B$  is larger than that of  $B'$  i.e.

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

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

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

We also have the following result, based on the definitions of the evaluation criteria in section 2.2.

**Behavior Evaluation Theorem** If  $B_1 >_p B_2$ , then  $B_1 >_u B_2$ ,  $B_1 \geq_f B_2$ , and  $B_2 \geq_m B_1$ .

## 5. Designing Behavior Sets

In this section, we show how the aforementioned criteria can be used to evaluate behavior sets and to improve them. Throughout this section, we assume that the state space of the environment including the robot is finite. Specifically, we first show how computations related to  $\tau_G(B)$ , usefulness, flexibility, modularity, and reliability can be carried out. Then, we show how any two behavior sets can be compared using these criteria. Finally, we report on guidelines for improving a given behavior set.

**Computing  $\tau_G(B)$ :** Let us consider the computation of  $|\tau_G(B)|$  given a behavior set  $B$ .  $|\tau_G(B)|$  is found by measuring the number of paths in the tree of world states such that the states are generated in a breadth-first fashion, starting from the initial state of world. Furthermore, no world state should appear more than once in the tree. The tree is grown until no new world states can be found. A node in the tree is the same as a world state. We will use the terms node and world state interchangeably. The initial state of the world is the root node. Each non-root node is generated by applying a behavior. The world state resulting from the application of a behavior can be found by algebraic manipulation of the previous world state and the consequence of the behavior. For example, let the initial world state be  $(a \wedge b \wedge c \wedge d)$ . Let  $B = \{\beta_1, \beta_2, \beta_3, \beta_4\}$  such that  $s_1 = (a \wedge b)$ ,  $c_1 = (\neg a \wedge e)$ ,  $s_2 = (a \wedge c)$ ,  $c_2 = (\neg a \wedge f)$ ,  $s_3 = (a \wedge d)$ ,  $c_3 = (\neg a \wedge g)$ ,  $s_4 = (e \wedge f)$ ,  $c_4 = (\neg e \wedge h)$ . In this case, three behaviors  $(\beta_1, \beta_2, \beta_3)$  are applied to the initial state in the process of generating the state tree. The three resulting world states are  $(b \wedge c \wedge d \wedge e)$ ,  $(b \wedge c \wedge d \wedge f)$ , and,  $(b \wedge c \wedge d \wedge g)$ . Generating the tree in a breadth-first fashion also ensures that each task-fulfilling chain found is the shortest of all chains fulfilling that particular task. It is not necessary to construct different trees for different initial states of the world. This is because most world states in any tree will be a part of any other different root node valued tree. The number of world states in any  $\tau_G(B)$  tree is bounded above by the product  $\prod_{i=1}^n s_i$ . Here  $n$  is the number of objects in the world. The number of distinct states of object  $i$  is  $s_i$ .

Construction of the tree representing  $\tau_G(B)$  is a one time effort. As behaviors are added or deleted, the tree can be incrementally updated without starting from scratch.

**Computing  $U(B)$ :** This is easily found once  $|\tau_G(B)|$  is found.

**Computing flexibility:** Flexibility is a relative criterion. The computation of  $\tau_G(B)$  provides information useful for comparing the flexibilities of two behavior sets. The lengths of the chains fulfilling each task in  $\tau_G(B)$  can be recorded. For example, in Fig. 3, as far as the task  $\langle I', G' \rangle$  is concerned, there are three world states in which  $I'$  is true and there are three world states in which  $G'$  is true. Hence, there are three chains fulfilling the task  $\langle I', G' \rangle$  and their lengths are 1,1 and 2.

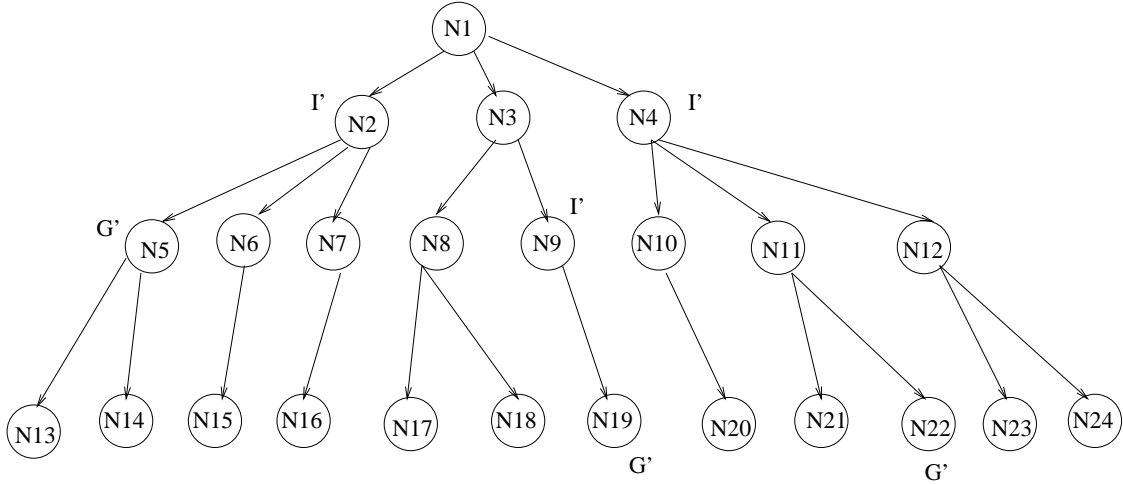


Figure 3. Tree representing  $\tau_G(B)$  for a small behavior set  $B$ .

**Computing Modularity:** Since the modularity of a behavior set  $B$  is the inverse of  $L_{cycle}(B)$ , we will show how  $L_{cycle}(B)$  is computed. For each behavior chain  $C$  in the tree representing  $\tau_G(B)$  count the number of pairs of behaviors  $\beta_i$  and  $\beta_j$  such that (i)  $\beta_i$  precedes  $\beta_j$ , and, (ii)  $c_j \Rightarrow s_i$ . Let the number of such pairs of behaviors be  $score(C)$ .  $L_{cycle}(B)$  is then  $\sum_{C \in \tau_G(B)} score(C)$ .

**Computing Reliability:** Reliability is a relative criterion. Still, the information in the  $\tau_G(B)$  tree can be used to compute the probabilities of fulfilling various tasks. Consider the tree from Fig. 3. It has 12 leaves. The nodes in a path from node  $N_i$  to any leaf are descendents of  $N_i$ . A node is not a descendent of itself. Nodes on a path from the root to a node  $N_i$  are ancestors of  $N_i$ . A node is not its own ancestor. Consider the task  $\langle I', G' \rangle$  to be fulfilled with a chain from the tree in Fig. 3. In the tree, there are three world states in which  $I'$  is true. They are  $N_2, N_9$ , and,  $N_4$ . Likewise, there are three world states in the tree in which  $G'$  is true. They are  $N_5, N_{19}$ , and,  $N_{22}$ .  $p(N_i, N_j)$  denotes the probability of reaching the state  $N_j$  from  $N_i$ .  $p(\langle I, G \rangle)$  denotes the probability of fulfilling the task  $\langle I, G \rangle$ . The probability of fulfilling a task is the minimum of the probabilities of executing all chains such that the executions all start in a state containing  $I'$  and all end in a state which contains  $G'$ . The probability of execution of a behavior chain

depends on the probabilities of executions of individual behaviors in the chain. Thus,  $p(\langle I', G' \rangle) = \min(p(N2, N5), p(N9, N19), p(N4, N22))$  where  $\min$  denotes the minimum of its arguments.  $s(Ni, j)$  denotes the successor of node  $Ni$  on the path to the node  $Nj$ , such that the successor is also a child of  $Ni$ .  $\text{par}(Ni)$  denotes the parent of the node  $Ni$ . Thus,  $p(Ni, Nj) = p(Ni, s(Ni, j)) \cdot p(s(Ni, j), s(s(Ni, j), j)) \dots p(\text{par}(Nj), Nj)$ . It is clear from Fig. 3 that  $p(N2, N5) = 0.33$ ,  $p(N9, N19) = 1$ , and,  $p(N4, N22) = (0.33) \cdot (0.5) = 0.165$ . These probabilities are computed assuming that all behaviors whose stimuli are true in a world state have an equal chance of being executed. Thus,  $p(\langle I', G' \rangle) = 0.165$ .

The computational methodology above can be used to compare two behavior sets using one or more of the usefulness, flexibility, reliability, and, modularity criteria. An algorithm for conducting such a comparison is given in [15]. A given behavior set  $B$  can be modified by including and/or removing behaviors. If behaviors are only to be removed, there are  $2^{|B|} - 1$  removal possibilities. If some behaviors from a behavior set  $B'$  are to be included in  $B$ , there are  $2^{|B'|} - 1$  inclusion possibilities, assuming that  $B \cap B' = \phi$ . If both addition and removal of behaviors are permitted, then there are  $(2^{|B|} - 1) \cdot (2^{|B'|} - 1)$  possibilities, assuming that  $B \cap B' = \phi$ . This is clearly very huge. An iterative algorithm is given in [15] which allows a user to select behaviors to be added or removed from a behavior set iteratively and which compares the new behavior set with the original one. The comparison algorithm from [?] is used for comparing the new behavior set with the original one. The iterative algorithm terminates when the improvement obtained over the original behavior set is acceptable to the user.

We report on several guidelines for modifying behavior sets to improve them with respect to various criteria next. These guidelines can be used as heuristics for controlling a search in the space of behavior sets. They can also be used to manually modify behavior sets without using the iterative algorithm for improvement in [15]. The guidelines are motivated by the definitions and computational methodologies of the evaluation criteria.

**Guideline 1:** Include behaviors with very weak stimuli or very strong consequences to increase the usefulness of a behavior set.

This is because the size of the  $\tau_G(B)$  tree is increased by more powerful behaviors and tree size affects the usefulness of a behavior set.

**Guideline 2:** Remove behaviors with very strong stimuli or very weak consequences in order to increase the usefulness of a behavior set.

This is because behaviors with very strong stimuli or very weak consequences are likely to be less powerful. Thus, removal of such behaviors is not likely to cause a significant reduction in the size of the  $\tau_G(B)$  tree.

**Guideline 3:** Remove a behavior with a strong consequence  $c_j$  that succeeds behaviors with stimuli implied by  $c_j$  in the  $\tau_G(B)$  tree.

This is for increasing the modularity of  $B$ . The motivation for this is that the removal of such behaviors reduces the number of potential cyclic conflicts.

The ancestors of a behavior  $\beta_i$  in the  $\tau_G(B)$  tree are behaviors that precede  $\beta_i$  in some chain in the tree. The descendants of a behavior  $\beta_i$  in the  $\tau_G(B)$  tree are the behaviors that occur after  $\beta_i$  in some chain in tree. For example, consider a  $\tau_G(B)$  tree with only two chains which are  $\{\beta_2 : \beta_4 : \beta_7 : \beta_1\}$  and  $\{\beta_3 : \beta_5 : \beta_4 : \beta_9\}$ . The descendants of

$\beta_4$  are  $\beta_7, \beta_1$ , and,  $\beta_9$ . The ancestors of  $\beta_4$  are  $\beta_2, \beta_3$  and  $\beta_5$ . The descendants of  $\beta_2$  are  $\beta_4, \beta_7$  and  $\beta_1$ .  $\beta_2$  has no ancestors.

**Guideline 4:** Remove a behavior  $\beta_j$  with (i) a large number of ancestors whose stimuli are implied by  $c_j$  in the  $\tau_G(B)$  tree and (ii) with very few descendants in the  $\tau_G(B)$  tree.

This is to increase the modularity of  $B$  without causing a high reduction in  $\tau_G(B)$ . This is a variation of guideline 3. If a behavior has very few ancestors and a large number of descendants, its removal can cause lot of reduction in usefulness. in modularity. Guideline 4 helps avoid this.

**Guideline 5:** Include the behavior  $\beta_j$  with the least number of ancestors whose stimuli are implied by  $c_j$  in the  $\tau_G(B \cup \{\beta_j\})$  tree, to increase usefulness without significantly reducing modularity.

This is similar to guideline 4.

**Guideline 6:** Remove a behavior with a strong consequence  $c_j$  that succeeds other behaviors with stimuli implied by  $c_j$  in the  $\tau_G(B)$  tree, if the depths of  $\beta_j$  and its aforementioned ancestors differ significantly.

This guideline is for reducing the number of behaviors participating in a cyclic conflict. The difference between the depths of such a  $\beta_j$  and its ancestors indicates the number of behaviors participating in the cycles. Cyclic conflicts containing a large number of behaviors are more undesirable because the execution times of such cyclic chains are high.

**Guideline 7:** Remove behaviors with a large number of occurrences and/or a large number of descendants in the  $\tau_G(B)$  tree to improve the reliability of the behavior set.

The reliability of a behavior set is improved if the probabilities of executing task-fulfilling chains are increased. Removal of behaviors can increase reliability of a behavior set by reducing the amount of branching. And, a high degree of branching decreases the probability of executing a task-fulfilling chain. Removal of behaviors, however, can also decrease the usefulness of a behavior set. Thus, we have the next guideline.

**Guideline 8:** Remove behaviors with weak consequences and strong stimuli that occur at shallower levels in the  $\tau_G(B)$  tree in order to increase reliability of the behavior set without causing a high reduction in its usefulness.

**Guideline 9:** Include a behavior in  $B$  which is equivalent to a chain of behaviors in  $B$ , in order to increase  $B$ 's flexibility.

For example, consider the following four behaviors which pick up a glass, deposit a glass on the floor, deposit a glass on a table, and push an empty table respectively: *pick\_glass*, *place\_glass\_floor*, *place\_glass\_table*, *push\_empty\_table*. A *wander* behavior allows the robot to wander until an interesting object is found. If it is desired that a table and a glass should be moved to a new location such that the glass should be on the table after it is moved, we will need the following chain of behaviors  $\{pick\_glass : wander : place\_glass\_floor : wander : push\_empty\_table : wander : pick\_glass : wander : place\_glass\_table\}$ . A behavior for pushing a table (even if it is not empty) is equivalent to this chain of 9 behaviors because it achieves the state transition achieved by the chain. If such a behavior is included in the behavior set, the length of the shortest chain for fulfilling the task decreases from 9 to 1.

The  $\tau_G(B)$  tree is a very useful representation. A user can examine it to identify chains that are undesirable and then remove such chains from the tree until the tree is expectable.

Next, meta-level control (if it is allowed) can be used to prohibit execution of any behavior chain that is not a part of the acceptable  $\tau_G(B)$  tree. Note that such a meta-level control will not be the same as a planner, since the control would not generate a sequence of behaviors to be executed.

## 6. A Case Study

Let us consider a behavior-based robot working in a kitchen. Let the robot's behavior set consist of the following eight behaviors. The stimuli and consequences of the various behaviors are shown in parentheses following the behaviors' names.  $\text{location}(x,y)$  means that the location of  $x$  is  $y$ .  $\text{in}(x,z)$  means that  $z$  is in  $x$ .

$\text{open\_fridge}$  (**stimulus:**  $(\text{fridge}(x) \wedge \text{closed}(x) \wedge \text{location}(x,y) \wedge \text{at\_robot}(y) \wedge \text{gripper\_free})$ ,  
**consequence:**  $(\text{open}(x) \wedge \neg \text{closed}(x))$ ),

$\text{pick\_dish}$  (**stimulus:**  $(\text{dish}(x) \wedge \text{gripper\_free} \wedge \text{location}(x,y) \wedge \text{at\_robot}(y))$ , **consequence:**  $(\text{has\_gripper}(x) \wedge \neg \text{gripper\_free})$ ),

$\text{place\_dish}$  (**stimulus:**  $(\text{dish}(x) \wedge \text{has\_gripper}(x) \wedge \text{at\_robot}(y))$ , **consequence:**  $(\neg \text{has\_gripper}(x) \wedge \text{gripper\_free} \wedge \text{location}(x,y))$ ),

$\text{move}(x,y)$  (**stimulus:**  $\text{at\_robot}(x)$ , **consequence:**  $(\text{at\_robot}(y) \wedge \neg \text{at\_robot}(x))$ ),

$\text{open\_tap}$  (**stimulus:**  $(\text{tap}(x) \wedge \text{location}(x,y) \wedge \text{at\_robot}(y) \wedge \text{closed}(x) \wedge \text{gripper\_free})$ ,  
**consequence:**  $(\text{open}(x) \wedge \neg \text{closed}(x))$ ),

$\text{close\_tap}$  (**stimulus:**  $(\text{tap}(x) \wedge \text{location}(x,y) \wedge \text{at\_robot}(y) \wedge \text{open}(x) \wedge \text{gripper\_free})$ ,  
**consequence:**  $(\text{closed}(x) \wedge \neg \text{open}(x))$ ),

$\text{pick\_food\_fridge}$  (**stimulus:**  $(\text{gripper\_free} \wedge \text{fridge}(x) \wedge \text{location}(x,y) \wedge \text{at\_robot}(y) \wedge \text{open}(x) \wedge \text{food}(z) \wedge \text{in}(x,z))$ ,

**consequence:**  $(\neg \text{gripper\_free} \wedge \text{has\_gripper}(z) \wedge \neg \text{in}(x,z))$ ),

$\text{wash\_dish}$  (**stimulus:**  $(\text{dish}(x) \wedge \text{has\_gripper}(x) \wedge \text{dirty}(x) \wedge \text{tap}(z) \wedge \text{open}(z) \wedge \text{location}(z,y) \wedge \text{at\_robot}(y))$ ,

**consequence:**  $(\text{clean}(x) \wedge \neg \text{dirty}(x))$ ),

It is assumed that the two dimensional workspace of the robot is divided into a finite number of locations. Each location is an area.  $\text{move}(x,y)$  is the behavior of moving from area  $x$  to area  $y$ . The predicate  $\text{at\_robot}(y)$  is true if the robot is at any point in the area that represents location  $y$ . It is assumed that once a robot is at a location any object in that location is within the robot's reach. The actual path followed by a robot from one location to another is irrelevant.  $x,z$ , and  $y$  in the definitions of behaviors are variables. For example, the predicate  $\text{fridge}(x)$  is evaluated to true if there is an object in the range of the robot's sensors which meets the definition of a fridge. Thus,  $\text{fridge}(x)$  can be interpreted as  $\exists x \text{fridge}(x)$  (which means that there exists some  $x$  such that  $x$  is a fridge). The behaviors' actions are clear from their names.

Let us study how the evaluation criteria, theorems and lemmas apply to this domain. For this some new behaviors are considered. Though their stimuli and consequences are not given, the stimuli and consequences of the eight behaviors described earlier give an idea of how these may be defined. The behavior for picking up a dish can be replaced by a single more powerful behavior whose stimulus is  $\exists x((\text{dish}(x) \vee \text{glass}(x)))$ . This increases the greatest potential task space while keeping the size of the behavior set constant. Hence, the usefulness of the behavior set increases. Sometimes glasses and dishes may be

picked from and deposited at different places on the same empty flat surface. This requires a chain of at least three behaviors, e.g. *pick\_glass*, *move(A, B)* and *place\_glass*, where *A* and *B* are old and new locations respectively. To reduce the lengths of these chains, one can add behaviors like *push\_glass* and *push\_dish*. This makes the augmented behavior set more flexible than the original behavior set, because the tasks of moving dishes and glasses can be fulfilled with a chain of length 1 rather than with chains of length 3. Furthermore, the two added behaviors of pushing glasses and dishes could be replaced by one more powerful pushing behavior whose stimulus contains  $((dish(x) \vee glass(x)))$ .

If one unions the set of the 8 behaviors at the beginning of this section with the new behavior set  $\{turn\_on\_oven, set\_oven\_temperature, turn\_off\_oven\}$ , then the greatest potential task space increases. This indicates that the two unioned behavior sets are scalable. This is because food from the refrigerator or freezer can be heated in the oven, making it ready to serve. However, there are more potential cyclic conflicts like opening and closing water taps, opening and closing refrigerator doors, and repeatedly turning the oven off and on. These lead to additional longer cycles, e.g. turning the tap on, opening the refrigerator door, opening the freezer, opening the oven, closing the oven, closing the freezer, closing the refrigerator door and then turning off the tap. This indicates that the modularity of the behavior set lowers when it is augmented with behaviors for using the oven. If a more powerful behavior set is augmented with the three oven-related behaviors, its modularity will be less than or equal to the modularity of a less powerful behavior set augmented with the same three behaviors.

## 7. Discussion

Though the autonomous agent paradigm has become extremely popular over the last sixteen years, since Brooks' 1986 paper, no criteria have been defined to compare or evaluate behavior sets. We defined five such criteria to evaluate and compare behavior sets. They are power, usefulness, modularity, reliability, and, flexibility. We reported the results of relations between behavior sets using these criteria. We also defined four types of conflicts that can occur among behaviors. We showed how computations needed for applying the criteria can be carried out. We also reported on guidelines to improve a behavior set with respect to the various criteria.

Our work can be extended in several directions in the future. We have assumed that the consequences of behaviors are certain. This may not always be the case. For example, the consequence of picking a block up may not maintain the fact that the block is gripped since the block may slip and fall down. This can be addressed by identifying various consequences of behaviors and assigning probabilities to them (such that they sum to 1). The notion of fulfilling a task with certainty could be then replaced by the notion of fulfilling a task with a probability greater than or equal to a certain threshold. In that case a behavior could be said to be more powerful if it has a logically weaker stimulus that needs to be true with a lower probability. A behavior could also be said to be more powerful if it results in a logically stronger consequence that holds with a higher probability. In case the probabilities are hard to obtain, one can replace a behavior with multiple potential consequences by multiple behaviors, each with a certain consequence. Our analysis could then be extended. For example, if a behavior  $\beta_1$  has three possible

consequences  $c$ ,  $d$ , and,  $(c \wedge e)$ , then one can replace  $\beta_1$  with three behaviors  $\beta_{11}$ ,  $\beta_{12}$ , and,  $\beta_{13}$  such that  $c_{11} = c$ ,  $c_{12} = d$ , and,  $c_{13} = (c \wedge e)$ . These three behaviors could then be used in the evaluation of the behavior set containing  $\beta_1$ .

## 8. Conclusion

Though the autonomous agent paradigm has become extremely popular over the last sixteen years, since Brooks' 1986 paper, no criteria were developed for evaluating an agent's behavior set. We defined five criteria to evaluate a behavior set or compare behavior sets. These are power, usefulness, modularity, flexibility, and, reliability.

Whatever the chosen representation for stimuli and consequences may be, it will be important to have behaviors of higher power. Such behaviors provide higher functionality by executing over a wider range of situations and by creating stronger consequences. It will be necessary for behaviors to be chained in order to fulfill more complex tasks. The amount of chaining will continue to be a measure of the usefulness of the behavior set. Though certain behavior chains may be task fulfilling, they cannot be arbitrarily long. The flexibility criterion captures this. Also, the behavior sets may need new behaviors if the user expects them to fulfill more goals. However, the augmentation should not weaken the original functionality. Hence, scalability is important. Though it is desirable to have highly powerful, useful, and flexible behavior sets, there should not be undesirable interactions among the behaviors. Hence, the modularity criterion is relevant. The utility of our criteria is independent of the representation of the behaviors, and the physical structure of a robot.

**Acknowledgement** I thank the anonymous reviewers for their constructive comments. This work is funded in part by NSF grant IIS-0119630. I thank Marvin Minsky, Amitabha Mukerjee, Erann Gat, Pattie Maes, Avi Kak, Anthony Barrett, B. Chandrasekaran, Robin Murphy, Douglas MacKenzie, Ronald Arkin, Henry Hexmoor, Tom Addis, Marc Pavicic, Matt Mason, Subbarao Kambhampati, and Alan Schultz for encouraging me to continue work on behavior-based robots and/or offering useful comments on my work. I thank my students Michael Girmscheid and Randy Ellis for useful discussions about robot behaviors.

## REFERENCES

1. Arkin, R. C., Behavior-Based Robot Navigation for Extended Domains, *Adaptive Behavior* 1(2) (1992) 201-225.
2. Arkin, Ronald C., Fujita Masahiro, Takagi Tsuyoshi and Hasegawa Rika, Ethological modeling and architecture for an entertainment robot, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Seoul, Korea, 2001, pp. 453-458.
3. Birk Andreas and Holger Kenn, 2001, An industrial application of behavior-oriented robotics, *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*, Seoul, Korea, 2001, pp. 749-754.
4. Brooks, R. A., A robust layered control system for a mobile robot, *IEEE journal on robotics and automation*, 2(1) (1986) 14-23.
5. Cameron Jonathan M., MacKenzie Douglas C., Ward Keith R., Arkin Ronald C.,

- and Book Wayne J., Reactive control for mobile manipulation, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Atlanta, GA, Vol. 3, 1993.
6. Chapman David, Planning for conjunctive goals, *Artificial Intelligence*, 32(3) (1987).
  7. Connell, J., Minimalist mobile robotics, A colony style architecture for an artificial creature, Academic press Inc., 1990.
  8. Desai Rajiv, Rosenberg Charles and Jones Joseph, Kaa: An autonomous serpentine robot uses behavior control, Proceedings of the conference on Intelligent Robots & Systems (IROS), Pittsburgh, PA, 1995, pp. 250-255.
  9. Emery Rosemary and Balch Tucker, Behavior-based control of a non-holonomic robot in pushing tasks, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Seoul, Korea, 2001, pp. 2381-2388.
  10. Endo Yoichiro and Arkin Ronald C., Implementing Tolman's schematic Sowbug: Behavior-based robotics in the 1930s, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Seoul, 2001, pp. 477-484.
  11. Kirsh, D., Today the earwig, tomorrow man? *Artificial Intelligence* 47(1-3) (1991) 161-184.
  12. MacKenzie Douglas C., and Arkin Ronald C., Behavior-based mobile manipulation for drum sampling, Proceedings of IEEE International Conference on Robotics and Automation (ICRA), Minneapolis, MN, 1996, pp. 2389-2395.
  13. Maes Pattie and Brooks Rodney, Learning to co-ordinate behaviors, Proceedings of the National Conference on Artificial Intelligence (AAAI), Boston, MA, 1990, pp. 796-802.
  14. Mahadevan, S., and J. Connell, Automatic programming of behavior-based robots using reinforcement learning, *Artificial Intelligence*, 55 (1992) 311-365.
  15. Mali, Amol, On the evaluation of behavior-based agency, Technical report, Department of Electrical Engineering and computer science, University of Wisconsin, Milwaukee, WI, July 2002.
  16. Miller David, Desai Rajiv, Gat Erann, Ivlev Robert and Loch John, Reactive navigation through rough terrain: Experimental results, Proceedings of the National Conference on Artificial Intelligence (AAAI), San Jose, CA, 1992, pp. 823-828.
  17. Montgomery James, Fagg Andrew and Bekey George, The USC AFV-I, *IEEE Expert*, 10(2) (1995).
  18. Reiter, R., 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, 1991, pp. 359-380.
  19. Vera, A. H., and Simon Herbert A., Situated Action: A Symbolic Interpretation, *Cognitive Science* 17 (1993) 7-48.
  20. Wettergreen David, Pangels Henning and Bares John, Behavior-based gait execution for the Dante-II walking robot, Proceedings of the conference on Intelligent Robots & Systems (IROS), Pittsburgh, PA, 1995, pp. 274-279.