Behavior Bounding: Toward Effective Comparisons of Agents & Humans*

Scott A. Wallace and John E. Laird
Artificial Intelligence Laboratory
University of Michigan
Ann Arbor, MI 48109

Abstract

In this paper, we examine methods for comparing human and agent behavior. The results of such a comparison can be used to validate a computer model of human behavior, score a Turning test, or guide an intelligent tutoring system. We introduce behavior bounding, an automated model-based approach for behavior comparison. We identify how this approach can be used with both human and agent behavior. We demonstrate that it requires minimal human effort to use, and that it is efficient when working with complex agents. Finally, we show empirical results indicating that this approach is effective at identifying behavioral problems in certain types of agents and that it has superior performance when compared against two benchmarks.

1 Introduction

Over the past twenty years, AI research has successfully demonstrated a number of techniques for constructing agents that exhibit intelligent behavior. Many applications have the additional requirement that the agent's behavior be consistent with that of a human expert. This is especially true for tasks in which the agent must simulate a human such as in training situations or in virtual social experiences such as on-line gaming.

For these tasks, the standard approach to developing expert level agents begins with knowledge acquisition. Unfortunately, knowledge acquisition is usually imperfect. As a result, significant resources must be spent on validation, which often requires both a knowledge engineer and domain expert to monitor the agent's behavior in a large number of test scenarios. In this paper, we present a method for automatically comparing two actors' behavior that could be used to overcome this validation bottleneck.

The potential uses of automated behavior comparison extend well beyond knowledge-base validation. For example, a generalized approach for comparing two actors' behavior can be used to objectively score a Turing test. A perfect result would be indicated if no detectable differences between the human and its computer counterpart were identified. In addition, automated behavior comparison can serve as the core of an intelligent tutoring system, where the roles are reversed. A novice human's behavior is compared to a computer agent's behavior which serves as a gold standard and information about the student's errors is used to guide the lesson. In all of these applications, the basic process for comparing behavior is identical. The differences only stem from the source of behavior (e.g. human or machine, expert or novice) and how the results of the comparison are used (to identify programming errors, to score a test or to guide a lesson). For simplicity and cohesiveness, this paper will focus on using behavior comparisons to aid the knowledge-base validation problem, but the discussion and results can also be applied to the other tasks as well.

*This work was supported by the Office of Naval Research, contract N61339-99-C-O104

2 Interactive Human-Level Agents

The need for behavior comparisons is particularly pronounced when the agent must masquerade as its human counterpart (the expert). These agents, which we term *interactive human-level agents,* are distinguished by two properties. First, the agent's performance is judged based on its ability to behave as the human expert would behave. Secondly, like humans themselves, interactive human-level agents must interact with an external environment in order to perform many of their tasks.

A good example of an interactive human-level agent is TacAir-Soar [Jones *et al,* 1999]. TacAir-Soar flics virtual military planes as part of a simulated training exercise. Teammates may be other TacAir-Soar agents or human counterparts. Because the agents are intended to model expert level behavior, it is not acceptable just to achieve the final states (e.g. shooting down the enemy planes). Instead, the agent must generate the same behavior as the expert. Meeting this requirement is challenging because the expert may perform the task differently on different occasions.

In the remainder of this paper, we first begin by examining a simple method of comparing a computer agent's behavior to a human expert's behavior. Deficiencies with this method lead us to examine more sophisticated model-based approaches. In Section 4 we summarize desirable features of such an approach. Then, beginning in Section 5, we present our method of model-based behavior comparison.

3 Toward Automated Behavior Comparison

Before two actors' behavior can be compared, the behavior must be represented in a form that can be processed by the comparison algorithm. We can do this most easily by storing individual instances of behavior, or behavior traces. A behavior trace is a sequence of tuples B — $((s,G,a)_0,(s,G,a)_1,\ldots,(s,G,A)_n)$ in which each tuple $(s,G,a)_i$ indicates the environmental state (s), the goals being pursued by the actor (G), and the action being performed (a). The state and action portion of the behavior trace can be captured by observing the actor perform the specified task. The actor's goals are necessary to disambiguate instances when different actions are performed in equivalent environmental states. Depending on whether the actor is human or computer agent, the actor may need to record how their goals change during the task.

A simple approach to comparing the actor's behavior can be performed with the following steps:

Acquire a set of behavior traces from the human expert and the agent for the specified task. These sets, H and A, represent the human expert's and agent's behavior respectively over a number of different trials.

Extract relevant features from the behavior traces. Some information gathered through observation may not be useful to detect errors. In this step, the salient features from the sets H and A are used to create two new sets of sequences H^* and A^*.

Compare each sequence $a \in A^*$, to the contents of H^* Compute the minimal number of edit operations (insert, delete, modify) that would be required to transform a into /i, where h is the sequence in H^* that is most similar to a. Each edit operation indicates a potential error.

Report all deviations (after removing any redundancies) between the human's and agent's behavior. This report summarizes all potential errors.

This simple approach performs a more detailed analysis of behavior than simply checking that the agent and the expert reach the same final (goal) state. In this way, the agent's externally observable behavior as well as some aspects of its internal reasoning process can be inspected to ensure that it is consistent with the human expert's. In addition, this methodology has the ability to identify a large number of possible errors because it has access to all the salient properties of the behavior trace.

However, this simple approach also suffers from a number of potentially serious flaws. First, the representation of the actors' behavior is a set of sequences extracted from the behavior traces. These sets grow as more observations are considered. Because interactive human-level agents can typically solve problems in a number of different ways, and because the environments they operate within are complex, it is likely that a very large number of observations will be required to adequately cover the actor's behavior. This problem is exacerbated by the fact that the sequential representation makes no assumption about how the actor's behavior might be constrained. Although this makes it possible to use this simple approach with any variety of behavior, it also makes it impossible to leverage regularities that might exist in large classes of goal directed tasks.

4 Model Based Approaches

To improve upon the simple approach for automated error detection described in Section 3, we propose a model-based approach to comparing actors' behavior. Central to any such approach are the properties of the behavioral model. Our choice is guided by the following requirements:

Low Complexity Unless the new model is significantly less complex than the agent's knowledge base, understanding the model and the behavior it represents is no easier than examining the knowledge base directly. For the model to be an asset, it must provide an adequately accurate representation of behavior while remaining easy to understand.

Low Effort We have argued that one of the main uses of the behavior comparison is to reduce the cost of validating a human-level agent. In order to accomplish this goal, the human effort required to build the behavioral models must remain low.

Compatibility Behavior comparison has a number of potential applications, but most rely on being able to examine both human and software agent behavior. Thus the representation must be limited to data that can be collected from either of these types of participants.

Efficiency Human-level agents operate in complex environments and may perform their tasks in a variety of different ways. To address this problem, a model may be built using observations of expert behavior. In this case, it must be possible to generate the model efficiently, even if many observations are required.

Efficacy Meeting the preceding requirements will come at a cost. Most likely this will be a decreased ability to distinguish between some types of behavioral deviations (potential errors). A good representation will nonetheless be able to identify a wide range of behavioral deviations that are likely to occur within the target environments and overlook meaningless differences.

Prior work in model-based diagnosis (e.g. [Lucas, 1998]) has examined how to detect errors given a model of correct behavior. In general, however, the models in these systems are relatively complicated and intended to identify problems with mechanical or solid state devices as opposed to software agents. However, one system, CLIPS-R [Murphy and Pazzani, 1994] was designed expressly for validating software agents.

In CLIPS-R, the behavior model consists of environmental constraints that must be met initially, as well as during and after task execution. In addition, the model can include a finite state machine which identifies acceptable sequences of actions pursued by the agent. Superficially, the requirements for the CLIPS-R approach seem relatively simple to meet. However, specifying this additional knowledge is a manual process that can significantly increase human effort and ironically can introduce a recursive validation problem for the constraints.

5 Behavior Bounding

As an improvement to CLIPS-R and to the simple method presented in Section 3, our approach to behavior comparison, called behavior bounding, automatically and efficiently builds concise models of both the human's and agent's behavior by examining behavior traces. The model of the expert's behavior is used to identify boundaries on acceptable behavior, and potential errors are reported by comparing the model of agent behavior to these boundaries.

5,1 A Hierarchical Model

The advantages of behavior bounding all stem from its representation of behavior. Behavior bounding is inspired by the hierarchical representations used in AND/OR trees, HTN planning [Erol *et al.,* 1994] and GOMS modeling [John and Kieras, 1996] to encode the variety of ways in which particular tasks can be accomplished.

The hierarchical behavior representation (HBR) used in our approach is illustrated in Figure 1A. The hierarchy is an AND/OR tree with binary temporal constraints representing the relationships between the actor's goals and actions. In this representation, internal nodes correspond to goals and leaves correspond to primitive actions. A node's children indicates the set of sub-goals or primitive actions that are relevant to accomplishing the specified goal. For example, in Figure 1A, the sub-goals Destroy-Lead and Destroy-Wingman are relevant for completing their parent goal, Engage-Enemy. The manner in which sub-goals should be used to achieve their parent goal is encoded by the parent's node-type constraint (AND vs OR) and the ordering constraints between sub-goals. In Figure 1A, AND and OR nodes are represented with ovals and rectangles respectively. Binary temporal constraints are represented with arrows between siblings. Thus, the hierarchy specifies that Engage-Enemy may be correctly accomplished by first accomplishing Destroy-Lead and then accomplishing Destroy-Wingman.

This model of behavior is clearly less complex than the agent's underlying knowledge base, indeed, it is likely to be less complex than the model used by CLIPS-R. Behavior bounding abstracts away internal data-structures the agent may use in problem solving that cannot be represented by the constraints in the hierarchy. This means, that the HBR alone could not be used to perform some basic tasks such as depth first search. This begs the question, if the agent's behavior can be represented using such a simple structure, why was it not programmed in this representation to begin with? The hypothesis here is not that this representation is sufficient to *completely* capture the agent's behavior. Most human-level agents do rely on intermediate data-structures that are not available through the environment or through the structure of the goal hierarchy. However, our hypothesis is that the representation provided by behavior bounding is sufficient to identify a large class of possible errors in agent behavior without sacrificing efficiency. Moreover, we believe that behavior bounding can also help identify potential problem spots in the agent's knowledge (e.g. a specific goal) even if an exact error cannot be identified.

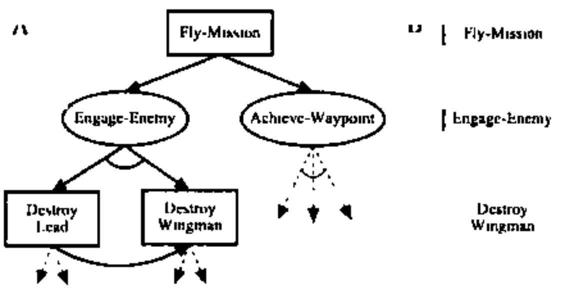

Figure 1: Hierarchical Behavior Representation & Goal Stack

In contrast to the behavior representations used for the simple comparison described in Section 3, the HBR makes two strong assumptions about the organization of the actors' knowledge and the effects this will have on their behavior. These assumptions increase the efficiency and efficacy of error detection for certain types of human-level agents.

The first assumption used by the behavior bounding approach is that the actor's goals are organized hierarchically, with more abstract goals placed toward the top of the tree. We also assume that at any point in the problem solving process the actor pursues a set of goals belonging to different levels in the hierarchy. This set, referred to as the goal stack, corresponds to a path in the hierarchy beginning at the top node and descending to the most concrete sub-goal that is currently being pursued by the actor. Figure IB illustrates a possible goal stack maintained by the actor whose behavior is represented in Figure 1 A.

The second assumption leveraged by behavior bounding relates to the independence of goals. Temporal constraints can only be formed between sibling nodes, and AND/OR classification determines which of a node's children must be performed for a particular task. This makes it is easy to constrain the way a particular goal is achieved, but difficult to represent constraints between arbitrary parts of the hierarchy. Although this may cause problems with some agent implementations, this property has significant benefits. Most importantly, it decreases the number of observations that are required. Consider a task that requires completing two goals, each of which could be fulfilled in four distinct ways. A sequential representation that makes no assumptions about goal independence (such as the one described in Section 3) would require sixteen distinct observations to cover the acceptable behavior space where as behavior bounding would only require four observations. This significant impact on efficiency is the direct result of leveraging the assumption about how goals are likely to add regular structure to an actor's behavior.

5.2 Identifying Errors

In general, we can view a behavior comparison method as an algorithm which divides the space of possible behaviors into two regions: behaviors that are likely to be consistent with the expert, and behaviors that are likely to be inconsistent with the expert. The simple comparison method described in Section 3 does this by enumerating consistent behaviors.

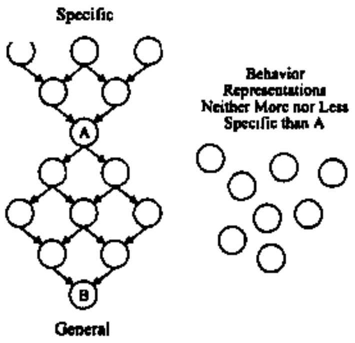
Specific
Behavior Representations Neither More nor Less Specific than A
A
B
General

Figure 2: Imposing Order on the Behavior Space

In behavior bounding, however, the constrained hierarchical representation allows us to break the space of possible behaviors into more refined regions.

We begin by noting that the constrained hierarchical representation allows us impose order on the space of possible behaviors. In particular, we can define an ordering from specific to general over the behavior hierarchies, by starting with a maximally constrained hierarchy (at the top) and iteratively removing constraints until none remain. Constructing a representation of an expert's behavior (Section 6) performs this same generalization, but most often stops before all constraints have been removed. Figure 2, in which each node represents a behavior hierarchy, illustrates this ordering. Once we have created a representation for the expert's behavior, we can identify the node it occupies in this ordered space (call this node A in Figure 2). This node (the upper boundary node) allows us to easily determine if the agent's behavior is likely to be correct. Because correct behavior must be consistent with expert behavior, an agent whose behavior representation is a specialization of the expert's (i.e. lies above A in the generalization lattice) exhibits behavior that is is likely to be correct.

The node that represents the completely unconstrained goal hierarchy is at the bottom of Figure 2 (labeled B) and provides a lower boundary. It contains the most basic specification for what may constitute acceptable agent behavior and as a result could be used to identify behavior representations that are known to be incorrect. Such representations would have a goal decomposition structure that was inconsistent with (i.e. contained different parent/child relationships than) this lower boundary (nodes in the right side of Figure 2).

Using the upper and lower boundaries described above we can classify any representation of agent behavior as: likely-correct (a specialization of the expert's behavior representation); likely-incorrect (a specialization of the expert's goal decomposition structure); and known-incorrect (inconsistent with the expert's goal decomposition structure).

Clearly, the hierarchical representation used by our approach describes behavior at a much higher level of abstraction than a typical knowledge base and in so doing, it presents a much more concise illustration of potential behavior than, for example, a set of individual rules. As a result, it meets our first requirement (low complexity). In addition, this representation can be generated automatically by examining behavior traces of an actor's performance on a task thus meeting the second requirement (low human effort). And because the behavior traces can be captured from either human or agent behavior with only minor support from the human participant, this model meets the third requirement. In the following sections, we will examine the remaining requirements in detail.

6 Learnability

In this section, we examine two aspects of behavior bounding's hierarchical representation: the effort required to create and maintain it, and its ability to represent behavior efficiently. Both of these requirements are addressed by the overall learnability of the representation. That is, if the representation can be learned from observations (as we have suggested), then it requires human effort only to initiate the learning process. If the learning procedure is efficient, and the data structure's growth is limited, we can further say that the hierarchy represents behavior efficiently.

The learning procedure for constructing the HBR extracts goal stacks and actions from a behavior trace, forming a hierarchical structure such as the one illustrated in the previous section. After processing the first behavior trace, the hierarchy contains the maximum number of constraints (i.e. AND/OR constraints on the goals and binary temporal constraints between siblings) that are consistent with the behavior in the trace. So, if each goal in the hierarchy is pursued only once while performing the task, all internal node-types are AND (maximally constrained) and all sibling internal nodes are totally ordered (again, maximally constrained).[1] Upon examining subsequent behavior traces, the hierarchy is generalized in such a way that it remains maximally constrained with respect to all of the behavior traces it has processed.

Due to page limitations, we cannot present the learning algorithm in detail, however it should be clear that the hierarchy can be built as described above with complexity $O(L|N|^2)$, where $|N|$ is the size of the goal hierarchy and L is the length of the behavior trace. In most cases, it is reasonable to assume that one property of expert quality behavior is completion of the task within a number of steps proportional to $|N|$. When this assumption holds, we can say that this algorithm is bounded by $O(|N|^3)$ in time and space, with respect to the size of the input (i.e. the length of the behavior trace). Because this complexity is a low order polynomial of $|N|$, the hierarchy is efficient when encoding an instance of behavior.

We can also classify the sample complexity of our hierarchical representation. We can think of our representation as an ordered tuple $P = (p_1, p_2, \ldots, p_{|N|})$ where each p_i is itself a tuple containing the type of the node z (either AND or OR), as well as a list $L = (l_1, l_2, \ldots, l_{|N|})$ such that $l_a = 1$ iff g_i is ordered before l_a Note that since ordering constraints only occur between siblings, the length of the list L would only need to be length $|N|$ in the degenerate case. The size of this hypothesis space is bounded by $2^{|N|+|N|^2}$. Using Haussler's equation [Haussler, 1988], the number of

[1]The leaves, representing primitive actions, will only be totally ordered if each action was used only a single time to achieve its parent goal.

training examples m required to learn the appropriate behavior representation is bounded by:

$$m \geq \frac{1}{\epsilon}\left((|N|^2 + |N|)\ln(2) + \ln\left(\frac{1}{\delta}\right)\right) \qquad (1)$$

This indicates that the required sample size is polynomial with respect to the number of goals in the hierarchy $(|N|)$. This, together with the fact that the time required to incorporate a new behavior trace into the learned HBR is also polynomial in $|N|$, shows that our representation is PAC-Learnable. This means that the HBR efficiently represents aggregate behavior as well an individual instance of behavior, thus meeting our fourth requirement.

7 Efficacy

The efficacy of behavior bounding is addressed by two components. First, how good is the unconstrained hierarchical representation (the lower boundary) at identifying behavior that is known to be incorrect. Second, how well does the expert's representation (the upper boundary) serve to distinguish between potentially correct and incorrect behavior.

At first glance, it is not obvious how much behavior can be filtered by the lower boundary. However, its effectiveness as a filter is quite surprising. Consider an unconstrained behavior representation with branching factor *b* and depth *d*. Without loss of generality, assume that the nodes are uniquely labeled. For simplicity, also assume that at any level in this hierarchy, the actor completes its current goal before starting the next goal. Then, we could define an actor's behavior as a sequence of symbols chosen from the lowest level of the unconstrained hierarchy. For behavior sequences of length b^d, in which no symbol is repeated, there are $b!^s|_{s=\sum_{j=0}^{d-1} b^j}$ possible sequences that are consistent with the goal decomposition of the unconstrained hierarchy. In contrast, there are $b^d!$ sequences in which the symbols may be placed without necessarily conforming to the unconstrained hierarchy. For a hierarchical structure of depth 4 and branching factor 2, only 1 in approximately $6.4 \cdot 10^8$ of the possible sequences of length 16 are consistent with the goal decomposition specified by the unconstrained hierarchy. This illustrates the potential power of the lower behavior boundary to discriminate between behavior that is potentially correct and the large collection of behavior that is inconsistent with the expert's goal decomposition structure, and thus known to be incorrect.

To examine how well the expert's representation distinguishes between correct and incorrect behavior we would ideally examine a large set of hand-programmed agents before they have been validated. Unfortunately, this is not feasible. Instead, we make random modifications to an agent's knowledge base. These modifications introduce unbiased behavioral flaws in the agent program, and experiments are performed to determine how well each type of error is identified. Because the modifications are made randomly, the errors that are examined will not be biased by our expectations about how easily they will be to identified.

Our experiments are performed on a series of agents within a simulated object-retrieval environment. The object-retrieval task requires both planning and reactive reasoning. Initially, when given the task, the agent must plan a route through known territory to a building thought to contain the desired artifact. Because the agent has no prior knowledge of the building's layout, it must explore the facility until the object is found, and then find its way back out. The task is complete once the agent leaves the building with the object. A behavior comparison metric's performance is judged based its ability to correctly identify errors in agent behavior, to identify *all* errors that have occurred, and to produce minimal amounts of spurious information in the report.

7.1 Methodology

We implemented the algorithm described in Section 6, along with two version of the simple approach described in Section 3 to serve as benchmarks. The first benchmark, the action sequence, extracts the sequence $A = (a_0, a_1, \ldots, a_n)$ from the behavior trace $B = ((s,G,a)_0, (s,G,a)_1, \ldots, (s,G,a)_n)$ while the second benchmark extracts the sequence of goals $G = (g_0, g_1, \ldots g_n)$ from *B*. Remember that the benchmarks are not particularly efficient representations; they can grow exponentially and have an exponential sample complexity. However, they do make interesting benchmarks of efficacy.

We initially constructed an agent that solves the problem in a very rigid manner. That is, across different attempts, the agent will complete the task using identical behavior so long as it is provided identical initial states and so long as the environment responds identically to its behavior. Given this agent, we performed modifications on its knowledge-base by randomly removing rules that determine preferences between competing goals and actions. The results of these modifications are agents that complete the task successfully in the traditional sense (i.e. they reach the same end state), but have increased flexibility in terms of the sequence of goals and actions they use to achieve that final state. In addition, the behavior exhibited by these modified agent's *cannot* be classified as incorrect by the lower behavior boundary node. As a result, these tests directly examine the abilities of the upper boundary node to distinguish between correct and incorrect behavior.

Each family of experiments begins by selecting two agents, e and n, such that n is a modified (more flexible) version of e. We designate e as the expert, and n as the novice. Because n is more flexible than e, it will behave in certain ways that are not consistent with expert behavior—these are errors.

After the expert and novice have been selected, they are individually incorporated into a simulation so their behavior can be observed. We then gather between 10 and 15 behavior traces of the actors performing their task, ensuring that no two behavior traces are identical. These traces form the sets BT_E and BT_N for the expert and novice respectively. Finally, each behavior trace in BT_N is examined manually to determine what errors it contains.

The captured behavior traces are then split into a number of subsets: $n_i \subset BT_N$ and $e_i \subset BT_E$. A single experiment consists of examining each comparison method's performance on a pair of these subsets $(n_i$ and $e_j)$. A *family* of experiments contains the experiments that compare all n_i to all e_j for a particular novice/expert pair. Thus comparing

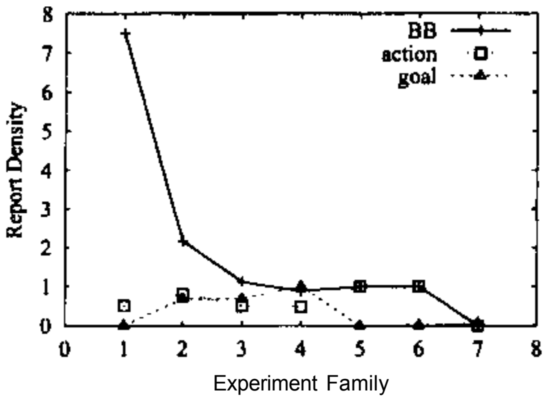

Figure 3: Report Density

four expert/novice pairs results in four experiment families although the total number of individual experiments may be much larger. By constructing experiment families in the way, we are able to examine the impact of different observational data more easily.

7.2 Results

For each experiment, we begin with a record (summarized by our manual examination of the agent's behavior traces) of what behavior errors were committed by the agent; these are true errors. This record identifies both low-level true errors such as omissions, commissions, and intrusions, as well as higher-level true errors such as misplacement, or repetition which are the result of multiple low-level errors. We then collect a summary from each comparison method about how the agent's behavior differed from the expert's behavior. Based on this, we record the number of true errors that occurred but were not identified by the summary (false negatives). Then, for each behavioral deviation in the summary we manually determine whether it indicates a new true error in the agent's behavior. If so, this deviation is meaningful and thus a useful part of the report, otherwise it is spurious.

To judge the performance of a comparison method, we first determine how many errors it fails to identity (false negatives). From this standpoint, all comparison methods performed relatively similarly. Across all experiment families the range on false negatives is between zero and 1.75 for all comparison metrics. However, in each experiment family behavior bounding does perform slightly better than the rest, maintaining the lowest average score, and achieving zero false negatives in five experiment families.

The other performance criteria we are interested in is how much of the comparison method's summary is useful. Because the summary is likely to be analyzed by a human, spurious errors are undesirable because they require human effort to examine and follow-up before they can be dismissed. If a summary is largely spurious errors, it is unlikely that anyone would actually care to read it. On the other hand, if the summary provides a large amount of good information, an occasional spurious error is likely to be acceptable.

This can be expressed as a ratio of how many true errors can be identified by the deviations listed in the summary to the total number of deviations listed in the summary. We call this value the report density of the summary. Thus, a comparison method that reports no meaningful deviations, and only spurious information will have a report density of zero. Note that it is possible for the report density to be higher than one. When a deviation in the summary correctly identifies a high-level error (such as the agent performing action *a before b* in contrast to the expert who performs *b before a)* we consider the summary as also identifying all the low-level errors that form this high-level error (e.g. two commissions in which *a* replaces *b* and vice-versa). This means that from the point of view of report density, it is better to identify high-level errors than low level errors. Because we should favor *concise* summaries, this is exactly what we want. Figure 3 illustrates the average report density of each comparison method's summary over all experiment families. Overall, behavior bounding outperforms the other metrics by achieving an report density of one or higher in six of the seven experiment families.

7.3 Conclusions

Our behavior bounding method uses an abstract model of behavior to identify potentially problematic differences between two actors. This in turn leads to a semi-automated method that can be used to validate complex, human-level behavior. Moreover, behavior bounding offers significant advantages over prior validation approaches and traditional manual techniques: it can be easily be used to validate an agent when human performance is the specification for acceptability; it requires only minimal human effort to initiate and perform the behavior comparisons, and behavior bounding is effective at identifying errors, even when compared to methods that are not constrained by efficiency requirements.

References

[Erol *et al,* 1994] K. Erol, J. Hendler, and D. S. Nau. HTN planning: Complexity and expressivity. In *Proc. of the \2th National Conf on Artificial Intelligence,* pages 1123-1128. AAAI Press/MIT Press, 1994.

[Haussler, 1988] D. Haussler. Quantifying inductive bias: AI learning algorithms and Valiant's learning framework. *Artificial Intelligence,* 36:177-221, 1988.

[John and Kieras, 1996] B. E. John and D. E. Kieras. The GOMS family of user interface analysis techniques: Comparison and contrast. *ACM Transactions on Computer-Human Interaction,* 3(4):320-351, 1996.

[Jones et al., 1999] R. M Jones, J. E. Laird, P. E. Nielsen, K. J. Coulter, P. Kenny, and F. V. Koss. Automated intelligent pilots for combat flight simulation. *AI Magazine,* 20(l):27-42, 1999.

[Lucas, 1998] P. Lucas. Analysis of notions of diagnosis. *Artificial Intelligence,* 105:295-343, 1998.

[Murphy and Pazzani, 1994] P. M. Murphy and M. J. Pazzani. Revision of production system rule-bases. In *Proc. of the 11th Int. Conf on Machine Learning,* pages 199-207. Morgan Kaufmann, 1994.