Specification Criticism via Policy-Directed Envisionment

Keith Downing
Stephen Fickas

CIS-TR-90-05
February 27, 1990

Area: Automated Reasoning
Topic: Software Development
Keywords: knowledge-based software development, critiquing, envisionment, qualitative physics

Abstract

Validating a complex system specification is a difficult problem. Generating behaviors and using them to critique a specification is one effective approach. Up until now, symbolic evaluation has been the key technique of behavior generation. Unfortunately, it has problems both in the amount of time it takes to complete a symbolic run, and in the large amount of uninteresting data it produces.

We propose policy-directed envisionment as an alternative to symbolic evaluation. This approach supplements the basic envisioning techniques of qualitative physics with behavioral goals in the form of *policies*. This combination overcomes the problems of symbolic evaluation by generating interpretations in a reasonable amount of time and by exploiting policies to prioritize and analyze the interpretations.

This paper describes the SC system, which employs policy-directed envisionment to critique specifications.

Department of Computer and Information Science
University of Oregon

Specification Criticism Via Policy-Directed Envisionment

Keith Downing
Stephen Fickas

Computer Science Department
University of Oregon
Eugene, Oregon 97403

Area: Automated Reasoning
Topic: Software Development

Keywords: knowledge-based software development, critiquing, envisionment, qualitative physics

Abstract

Validating a complex system specification is a difficult problem. Generating behaviors and using them to critique a specification is one effective approach. Up until now, symbolic evaluation has been the key technique of behavior generation. Unfortunately, it has problems both in the amount of time it takes to complete a symbolic run, and in the large amount of uninteresting data it produces.

We propose policy-directed envisionment as an alternative to symbolic evaluation. This approach supplements the basic envisioning techniques of qualitative physics with behavioral goals in the form of *policies*. This combination overcomes the problems of symbolic evaluation by generating interpretations in a reasonable amount of time and by exploiting policies to prioritize and analyze the interpretations.

This paper describes the SC system, which employs policy-directed envisionment to critique specifications.

I. Introduction

One of the major problems in building formal, operational specifications for complex systems* is accounting for the plethora of behaviors and behavioral interactions that might occur during the operation of those systems. Without some feeling for the space of possible behaviors, a specification designer can have little confidence in the robustness of the final system. Thus, to properly critique or validate a specification, one must have the capability to generate a fairly complete set of possible behaviors from it.

Up until now, the general approach to behavior generation for system specifications has been symbolic evaluation (see, for instance, (Cohen, 1983)). Using this approach, the user selects some set of specification actions to perform and an abstract description of an initial state. The symbolic evaluator then churns out behavior descriptions, often on a massive scale.

There have been two advances in the symbolic evaluation area. First, Fickas&Nagarajan (1988) showed that a case-based reasoning (CBR) approach could be used as a front-end to the symbolic evaluation process. In particular, their critic cataloged important cases to consider in a resource management domain, and fed these to a symbolic evaluator, thus partially automating the test-selection process. Full automation rests on two important assumptions: 1) that enough interesting test cases can be captured and stored in a case-base to confidently cover a domain, and 2) that one can match cases against complex specifications. The latter assumption, in particular, is problematic for any automated tool: it can be mapped to the AI-complete problem of concept recognition.

The second advance was directed to the back-end process of interpreting the results of symbolic evaluation. Swartout (1983) built a behavior explainer that took the raw behavior descriptions from a symbolic evaluator, and used heuristics to 1) look for interesting results, and 2) present them to the user in an understandable manner.

The key technique (and drawback) shared by both the critic and the behavior explainer is a reliance on the raw specification as a basis for evaluation and reasoning. Unless test cases and problem domains are tightly constrained, operational specification languages are much too complex a representation for automated behavior generation and critiquing tools. Of course the paradox here is that the more tightly we constrain our test data and problem description so to use existing symbolic evaluators, the less "symbolic" or abstract becomes the evaluation and its results. As an alternative, we have turned to abstract qualitative representations as a means of generating and interpreting specification behaviors.

The field of qualitative physics(QP) offers considerable assistance in this endeavor. Typically, qualitative simulators such as QPE (Forbus, 1986), QSIM (Kuipers, 1986) and QUAL (de Kleer, 1979) take qualitative constraints and a behavioral perturbation as inputs; they then derive a set of mutually-exclusive possible behavioral sequences (i.e. *interpretations*). Due to the ambiguities of qualitative arithmetic and local constraint propagation, this process, called *envisionment*, frequently produces a

* We define complex systems as those involving physical sub-systems, social sub-systems, and software sub-systems. In essence, we consider problems where a software system is embedded in an environment that includes physical and social components.

great many interpretations, some of which have a much higher likelihood of occurrence than others. Consequently, envisionment trades off accuracy for completeness by producing a comprehensive set of interpretations but having little knowledge as to which one might actually occur.

This suggests that if we can express a specification as a set of qualitative constraints, then envisioning techniques could be employed by a specification critic to elicit interpretations that describe the possible event sequences that the specification would permit. Of course, this also means that the specification critic will have little information as to which of those sequences will most likely occur. We must look back to qualitative physics for possible remedies for the accuracy-completeness tradeoff.

Recently, qualitative physics researchers have used everything from range arithmetic (Kuipers and Berleant, 1988) to high-order qualitative derivatives (de Kleer and Bobrow, 1984; Kuipers and Chiu, 1987) to phase spaces (Struss, 1988; Lee and Kuipers, 1988) in order to prune the interpretations down to a small set of most-probable behaviors. Basically, these efforts have supplemented envisioners with qualitative versions of sophisticated quantitative analysis tools. These tools have sufficient domain dependence (i.e. physical systems) to be of little use to the AI community as a whole. However, de Kleer (1979) has shown that knowledge of the purpose, function or *teleology* of a system can also significantly restrict the interpretation set; and the synonymy of 'teleology' with 'goal' makes de Kleer's work of widespread relevance to AI research - in particular, the type of specification criticism embraced by this project.

De Kleer's QUAL system (1979) uses local and global teleologies to filter the results of envisioning electrical circuits. For each interpretation, QUAL employs local teleologies to classify component behaviors, and global purposes to parse the local teleologies into a gestalt picture of the functioning device. QUAL prunes every interpretation that fails, at either the local or global level, to match any of the known teleologies. Since many of the possible but improbable interpretations have no known teleological import (relative to the standard teleologies of electrical engineering), teleology forms a strong bias for pruning the interpretation space.

Our work in specification criticism also involves teleological or goal-like constructs called *policies*. These represent biases as to how the policy-maker would like the system to behave. So if we run the specification through an envisioner, policies could help select the most desirable interpretation(s). Then, the critic could note the favored interpretations and take steps to insure that in the final design, only their behavioral sequences could occur. Thus, while QUAL takes an abstract/qualitative model of a completed design and attempts to recognize its correct behavior via teleology matching; the specification critic we investigate here would take an abstract model of an unrefined design and use envisionment followed by policy-based filtering to help clarify the design itself.

To investigate this integration of qualitative physics and specification criticism, we have designed SC, a specification critic based on policy-driven envisionment and interpretation analysis. The following section discusses SC's general operation along with its application to the domain of automated library systems.

II. SC - The Specification Critic

The general activity of SC closely mirrors that of a qualitative simulator: it inputs a set of constraints along with a behavioral perturbation and outputs a set of interpretations denoting the possible combinations of changes that the original perturbation could cause. However, unlike most qualitative simulators, SC runs under constant awareness of the set of active goals/policies. These policies compile into desired behaviors of certain local variables. For instance, in the library domain, the policy of keeping many books on the shelves compiles into an upper bound on the number of books that any one person can check out; and in the perturbation-based model of SC, this local goal is expressed as a recommendation that the individual check-out limit should never rise.

During envisionment, if the propagation of changes dictates that exactly one of n possible changes must happen next, and one of those is a rise in the check-out limit, then all of the other n-1 possible changes (that do not violate a policy) are marked as more desirable, since they prevent the check-out limit from rising and effectively help *preserve* the policy of well-stocked library shelves. For instance, in a university library, the addition of more, easier-to-use and/or cheaper copying machines might abate any need for higher check-out limits caused by a campus-wide increase in course workload.

When envisionment ends, the favored interpretations are those that violate the fewest policies; and the design recommendations are to insure that the desirable policy-preserving events of those interpretations actually do occur. These recommendations take the form of simple rules such as, "If there is a possibility of a global increase in course workload, and if the library seeks to maintain full shelves, then copying resources should be improved to alleviate the need for higher check-out limits."

III. Using The Specification Critic in the Library Domain

As a more detailed example, consider the library constraints of Figure 1. In this hierarchy, each node represents a variable, and constraints are formed by setting each parent node equal to the sum of its children. For instance, Active-Set = In-house + Checked-out, while In-house = Unwanted + Gain-Info. This tree represents some of the things that may happen to a library book. At the top, the active set is the complete set of books owned by the library. At any one time, a member of this set is either checked out or in the library (in-house). On the right side of the tree, a book may be checked out by a faculty member, graduate student or undergraduate. On the left side of the tree, an in-house book may be unwanted, or someone may seek to gain information from the book. To gain information, a library patron may either physically browse the book or utilize an on-line preview system. If browsed, the book may be simply reshelved, or the patron may seek to take home some information from the book. In this model, there are three ways to take home information: copy parts of the book, rip pages from the book, or steal the book.

III.1 Applying Confluence Theory to Library Models

To prepare this model for the perturbation analysis indigenous to envisionment, we simply rewrite each rule as a qualitative partial-differential equation, or *confluence* (de Kleer and Brown, 1985). For instance, the top constraint becomes ∂Active-set = ∂In-house + ∂Checked-out. That is, the qualitative change in the size of the active set equals the sums of the qualitative changes in the sizes of the in-house and checked-out collections. Now, for any variable X, ∂X can take on one of three values: [+] for 'rising', [-] for 'falling', and [0] for 'unchanging'.

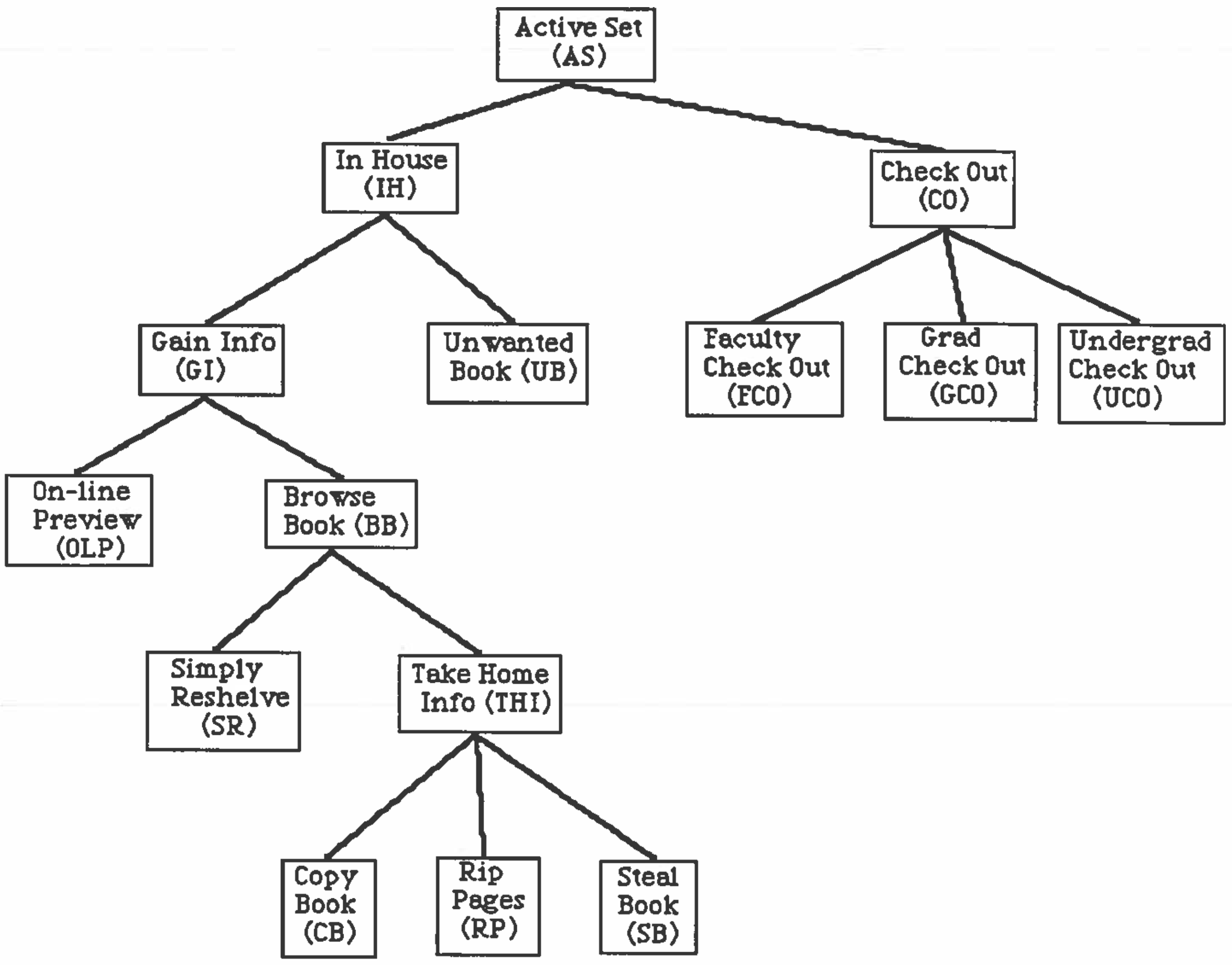

Figure 1: The Library Constraint Tree

During perturbation analysis, SC propagates changes through confluences via simple qualitative reasoning. First of all, we make an assumption analogous to de Kleer and Brown's (1985) confluence heuristic, which we call the "Most-Significant Change" Rule (MSC):

> When one variable of a confluence changes, it will cause a significant change in exactly one other variable of that confluence, while all other variables will remain constant.

SC makes no assumptions about **which** of the other variables will change, so it investigates all possible changes that satisfy MSC. Each such possibility (i.e. point of ambiguity) adds one or more additional interpretations to the final set produced by envisionment.

For instance, if ∂In-house = [+], then, under MSC, either:

1) ∂Active-Set =[0] and ∂Checked-out = [-], or
2) ∂Active-Set =[+] and ∂Checked-out = [0].

Verbally, if the number of in-house books rises, then either the cardinality of the active-set must also rise, and the number of checked-out books will hold steady; or, the active set will remain constant and check outs will decrease. Continuing the propagation of change, if check outs rise, then either faculty, graduate students or undergraduates will experience a significant rise in check outs; and again, SC considers each possibility in turn.

III.2 Policy-Directed Envisionment in SC

As mentioned earlier, SC differs from most envisioners by paying attention to policy-dictated variable-change preferences. In the library model, we employ a variety of policies such as "Good Condition Books", "Plentiful Stacks", and "Disseminate Information". These compile into local behavioral recommendations. For example, "Good Condition Books" requests that, whatever perturbations occur, the number of books with ripped pages should not increase. "Plentiful Stacks" demands that the set of in-house books should never decrease, and the number of stolen books should not increase. Finally, "Disseminate Information" embodies a goal of supplying as much information as possible to the library patrons, so it condemns a decrease in either the active set or the 'gain-info' books.

During constraint propagation, any changes that could violate an active policy are recorded as such. Furthermore, the alternate behaviors within any confluence that contains those violated variables are labelled as *policy-preservation acts*. For instance, if on-line previewing capabilities were to decrease, then two (mutually-exclusive via MSC) possible changes could occur: either the gain-info books would decrease, or the physically browsed books would increase. Now, if "Disseminate Information" were an active policy, then the decrease in gain-info books would be undesirable. Hence, an increase in browsed books would represent a policy-preservation act, since it could inhibit the gain-info decrease.

SC continues propagating changes to yield a complete set of interpretations (Table 1), which represent all possible (relative to MSC) consequences of decreasing on-line previewing. These interpretations appear in ascending order according to the number of interpretation behaviors that violate an active policy. In this example, the only active policy is "Disseminate Information". Values in the table represent the qualitative derivatives of the library variables. Starred changes denote policy-preservation acts, while those in shadow font (e.g. [-]) signify policy violations; an empty spot symbolizes no change.

	AS	CO	FCO	GCO	UCO	IH	GI	UB	OLP	BB	SR	THI	CB	RP	SB
1 *							AC*		[-]						
2 *									[-]	[+]*	[+]				
3 *									[-]	[+]*		[+]	[+]		
4 *									[-]	[+]*		[+]		[+]	
5 *									[-]	[+]*		[+]			[+]
6	AC*					[-]	[-]		[-]						
7		[+]*	[+]			[-]	[-]		[-]						
8		[+]*		[+]		[-]	[-]		[-]						
9		[+]*			[+]	[-]	[-]		[-]						
10							[-]	[+]	[-]						
11	[-]						[-]		[-]						

Table 1

The first five interpretations contain no policy violations and therefore represent the most desirable possible outcomes. The pivotal point in each of those interpretations is the use of a policy-preservation act to prevent 'gain-info' (GI) from falling. In 2 - 5, that act is an increase in the number of physically browsed books (BB), while in 1, SC makes an 'add-child' (AC) recommendation. That is, to prevent GI from declining, simply add another mechanism for gaining information (such as microfilm) and list it as a child of GI in the constraint hierarchy.

SC combines these policy-preservation acts into the following design recommendation:

> In the event of an on-line previewing decrease, be prepared to either increase the ease of physically browsing books or to add another means of gaining information from them so as to avoid a decline in the library's overall ability to disseminate information.

So, just as teleologies govern interpretation selection in QUAL, policies guide this selection in SC, along with pointing out the most salient behaviors.

By biasing interpretation selection according to the disseminate-information policy, SC essentially takes a library patron's point of view. Conversely, a library administrator may have other priorities, such as good-condition books and plentiful stacks. By using this duo as the active policy list, SC generates a different interpretation table:

	AS	CO	FCO	GCO	UCO	IH	GI	UB	OLP	BB	SR	THI	CB	RP	SB
1 *						AC*	[-]		[-]						
2 *							[-]	[+]*	[-]						
3 *									[-]	[+]	[+]				

4 *								[-]	[+]	[+] AC*			
5 *								[-]	[+]	[+]	[+]*		
6	[-]					[-]	[-]	[-]					
7		[+]	[+]			[-]	[-]	[-]					
8		[+]		[+]		[-]	[-]	[-]					
9		[+]			[+]	[-]	[-]	[-]					
10								[-]	[+]	[+]			[+]
11								[-]	[+]	[+]		[+]	

Table 2

Again, SC's envisionment produces eleven interpretations, five of which violate no policies. Unlike the first envisionment, this one employs many different policy-preservation acts among the highest-ranking interpretations. The first recommends the addition of another subordinate to 'In-house' in order to prevent a decrease in IH books, while the second prefers an increase in the unwanted books (UB) as a remedy. Neither of these is very informative and would surely be pruned if SC possessed meta-knowledge about its constraints. Interpretation three has no preservation acts, while four and five recommend changes to block an increase in either stolen books (SB) or page ripping (RP). To wit, #4 prescribes the addition of another method for taking home information (THI), while #5 advocates an increase in copying capabilities (CB).

Thus, policy decisions have considerable influence upon the generation and prioritization of behavioral interpretations. In addition, policies highlight the critical behaviors within any interpretation to isolate the pivotal constraints within a specification. Furthermore, policy-directed envisionment can lead to suggestions for supplementing those constraints with additional variables (as a means of preserving policies). All in all, policy-directed envisionment provides the completeness to explore the breadth of a specification's consequences, and the bias to evaluate and understand those behaviors.

IV. Related work

As shown above, SC borrows a few basic tools from qualitative physics: confluence theory and envisionment, which it then fortifies with policies to produce a qualitative simulator for social domains. In so doing, SC indicates the extensibility of QP techniques to non-physical situations; but most importantly, it illustrates the utility of qualitative, policy-directed simulation in specification development.

Looking to the knowledge-based software development field, Swartout's Gist Behavior Explainer, or GBE for short, (1983) comes closest to the research goals of SC. As with SC, GBE attempts to come to grips with the massive number of interpretations that can be generated from complex specifications. In SC this is handled by abstraction, in GBE by heuristic pruning and a reliance on the user to constrain test cases.

As with SC, GBE looks for interesting behaviors to present to a user. In GBE, interestingness is based on the domain-*independent* language Gist, and hence must center on features of the language rather than features of the domain. In SC, interestingness is also based on a domain-independent language, but that language is tied to domain-*dependent* policies: the focus of interpretation presentation is on their preservation. Related to this, we note an interesting insight by Swartout in his future work section (1983):

> The current [Gist] symbolic evaluator is not goal driven. Rather than having a model of what might be interesting to look for in a specification, the evaluator basically does forward-chaining reasoning until it reaches some heuristic cutoffs...By giving it, at least at a high-level, a model of what might be interesting, it could be more directed in its search. After narrowing the search using goals, the evaluator could then switch to forward-chaining to more completely examine the smaller problem space.

It is not hard to view SC as a front-end process to a full symbolic evaluator such as that used by the GBE. By joining the two, we would rely on SC to narrow an area of concern, and then call on a more brute-force, symbolic evaluation approach to explore details.

Finally we note that SC will attempt to fill in missing portions of a specification to avoid producing an interpretation that violates a policy (the ACs of tables 1 and 2). The DESIGNER system takes a similar action when faced with a deficiency in an algorithm under design (Steir and Kant, 1985).

References

Cohen, D. (1983). Symbolic execution of the Gist specification language, In *Proceedings of the 8th International Joint Conference on AI.*

De Kleer, J. (1979). *Causal and Teleological Reasoning in Circuit Recognition* (Lab Rep. No. 529). MIT AI Lab.

De Kleer, J. and Bobrow, D. (1984). Higher-Order Qualitative Derivatives. Proceedings AAAI-84, Austin, Texas (pp. 86-91).

De Kleer, J. & Brown, J. (1985). A Qualitative Physics Based on Confluences. In D. Bobrow (Ed.), *Qualitative Reasoning about Physical Systems* (pp. 7-84). Amsterdam, The Netherlands: Elsevier Science Publishers B.V.

Fickas, S., Nagarajan, P., (1988). Being suspicious: critiquing problem specifications, In *Proceedings of the 1988 AAAI Conference*, Minneapolis, Minn.

Forbus, K. (1986). *The Qualitative Process Engine* (Tech. Rep. No. 1288). Urbana-Champaign, Illinois: Univ. of Illinois.

Kuipers, B. (1986). Qualitative Simulation. *Artificial Intelligence*, 29 (3), 289-338.

Kuipers, B. & Berleant, D. (1988). Using Incomplete Quantitative Knowledge in Qualitative Reasoning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 324-329). St. Paul, Minnesota: Morgan Kaufmann Publishers, Inc.

Kuipers, B. and Chiu, C. (1987). Taming Intractible Branching in Qualitative Simulation. *Proceedings of The Tenth International Joint Conference on Artificial Intelligence.* Milano, Italy: Morgan Kaufmann Publishers, Inc.

Lee, W. & Kuipers, B. (1988). Non-Intersection of Trajectories in Qualitative Phase Space: A Global Constraint for Qualitative Simulation. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 286-290). St. Paul, Minnesota: Morgan Kaufmann Publishers, Inc.

Steier, D., Kant, E. (1985). The Roles of Execution and Analysis in Algorithm Design, In *IEEE Transactions on Software Engineering*, Vol. 11, No. 11.

Struss, P. (1988). Global Filters of Qualitative Behaviors. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 275-279). St. Paul, Minnesota: Morgan Kaufmann Publishers, Inc.

Swartout, W. (1983) The Gist behavior explainer, In Proceedings of the National Conference on AI.