Supporting self-explanation on behavior-based robots

Christopher Dac Le

Computer Science Dept/Northwestern University
1890 Maple Ave, 3rd floor
Evanston, IL 60201
dac@cs.northwestern.edu

Abstract

The **RO**bot **S**elf-**E**xplains wh**Y** (ROSEY) project is an attempt to extend behavior-based robots by supporting verbalization of causal explanations of the robot's own behavior. It extends parallel-reactive architectures to support higher-level functions, in this case self-inspection and explanation. In this paper, we discuss work in progress on generating explanations of locomotive behaviors and describe our initial implementation on an indoor mobile robot.

Introduction

The **RO**bot **S**elf-**E**xplains wh**Y** (ROSEY) project attempts to demonstrate how a behavior-based robot can verbalize causal explanations of its own behavior. More generally, it attempts to show how a behavior-based system can carry out a higher-level task such as explanation.

Traditional behavior-based systems demonstrate limited high-level reasoning. Since they are implemented as a fixed network of wires, they cannot handle variable binding and are forced to use simple representations that are effectively limited to propositional logic. Exhibiting higher-level functions such as meta-level reasoning is therefore more difficult on behavior-based systems than on symbolic systems.

However, we believe behavior-based design need not preclude these abilities. ROSEY shows how a behavior-based system can obtain *reflective knowledge* of itself by inspecting and reasoning about its own internal processes and state in order to generate an explanation.

We begin by sketching a design for the explanation task. We describe an explanation process that is based on examining the structure of the robot's program rather than based on accessing a runtime diagnostic system. We then describe the mechanisms required for supporting explanation. Finally, we describe work-in-progress on the initial implementation and close with preliminary results.

Our initial goal is to build a minimalist system. It is neither to advocate a specific cognitive theory nor to evaluate self-explanation as a human-robot interface paradigm. Although we plan on pursuing the latter, we presently ignore issues such as the need to tailor a robot's explanations to the target user; instead, we currently assume that the robot's target user is its programmer.

A process for explaining behavior

Most behavior-based systems can be described as circuits: signals flow from sensors through wires and processing elements to the control inputs of effectors. We want ROSEY to answer questions about its own behavior, for example "`Why are you turning?`" Since behavior-based systems are circuit-like, explanation can be reduced to tracing signal flow: start at the signal controlling the motors, and trace backwards through the circuit, discussing the active elements that drive it. This approach presumes that the explanation component can inspect the relevant circuit structure at runtime.

Tracing causality

The problem however, is that the circuit tends to be complicated. Others have raised similar issues related to the automatic explanation of detailed process models (Williams 1991; Nayak 1992; Mallory, Porter, and Kuipers 1996). If every traced element is discussed, then explanations will become overly detailed and more difficult to grasp. To avoid this problem, we generate descriptions in terms of an *abstract* circuit in which whole sub-circuits are collapsed to single, idealized nodes. In the case of ROSEY, we abstractly describe the circuit in terms of *propositions* (e.g. behavioral states such as "I am turning") and *operators* (e.g. behaviors, plans).

This simplified representation retains properties of the complete circuit that an explanation system can still use to answer the following questions:

- ***What behaviors must be concurrently executing in order to yield the observed actuator behavior?*** We define operators as both the internal and external behaviors that the robot may execute during any single execution cycle. Thus, we want to include behaviors that control the robot's actuators, as well as the cascade of internal behaviors that drive the activation of the current behavior.

- ***What set of conditions must hold true in order to yield the observed actuator behavior?*** We treat propositions as corresponding to internal conditions that the robot may have during any single execution cycle. Propositions encompass sensory-motor conditions as well as the Boolean conditions under which an operator will be activated.

Figure 1 illustrates how propositions and operators are causally related: propositions affect operators that then effect other propositions. In other words, operators can change the truth-values of propositions, whereas propositions may directly turn operators on or off. Tracing causality can be reduced to a process of walking this circuit (which we call the *abstract causal circuit*) and finding the path of activated propositions and operators. We currently assume only one active path exists through this circuit.

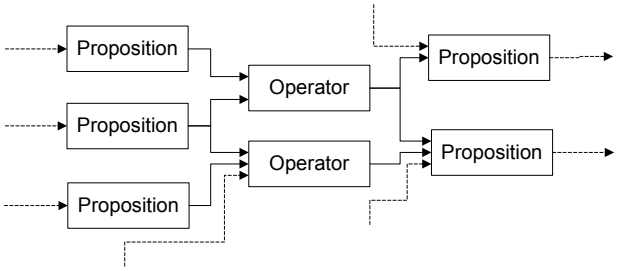

Figure 1. Abstract causal circuit. *A robot circuit is simplified to nodes consisting of either propositions or operators.*

However, we still need to formulate how to actually generate this structure given the original robot circuit. In particular, how do we identify a sufficient set of propositions and operators to encode? What is the process for mapping the elements of a behavior-based system to this set? We are currently investigating possible answers to these questions. For the moment, we are manually constructing the abstract causal circuit.

Self-inspection mechanisms

The robot will require some runtime mechanisms to operate over the abstract causal circuit. One is the means to look up and tag an arbitrary node, given its lexical equivalent. For example, the word "turning" should correspond to a particular node in the circuit.

Another set of needed mechanisms involves the extraction of both static and dynamic information pertaining to the node. Required information includes:

- Name
- Type (proposition or operator)
- Current state
    - If node is a proposition, is it currently true?
    - If node is an operator, is it currently running?
- Possible causes
    - If node is a proposition, what operators are known to cause it to be true?
    - If node is an operator, what propositions are known to activate it?
- The active cause
    - If node is a proposition, which operator is actually causing it to become true?
    - If node is an operator, which proposition is actually activating it?

These requirements directly translate into a corresponding set of operations and a simple interface for retrieving this information from the self-inspection (SI) system:

- `get-name`
- `get-type`
- `get-current-state`
- `get-known-causes`
- `compute-active-cause`

A simplified explanation process

Given these mechanisms, we can now sketch a process for generating explanations. Figure 2 illustrates a schematic overview of this process. To initiate an explanation, the user asks a "why" question resembling any of the following forms:

- "Why are you <symptom>?"
- "Why are you in the state of <symptom>?"

Here, <symptom> is a word describing a particular robot state (e.g. "turning" or "moving") and corresponding to a proposition node.

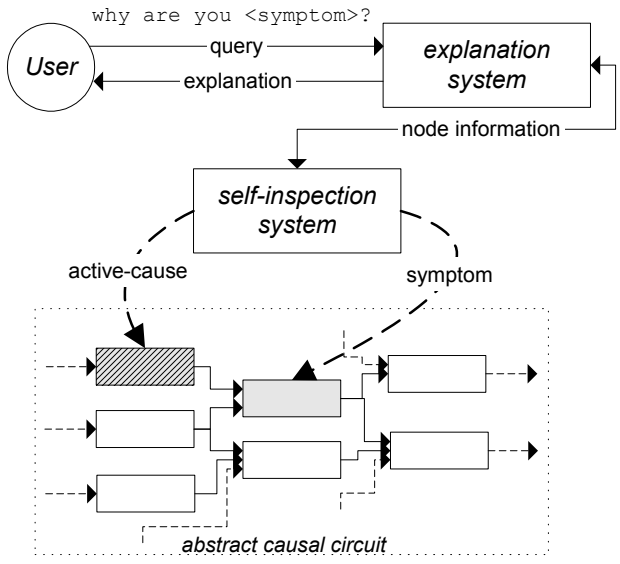

Figure 2. Schematic overview of the explanation process.

The explanation system parses the user's question. Upon recognizing a why question, the explanation system tags the word representing the symptom so that the SI system may look up the corresponding node; the SI system references this node as the "symptom."

At this point, the SI system tests whether the symptom node is even active, in other words, whether the symptom is even true. By testing the truth-value of the symptom, the explanation system can comment on the applicability of the user's question. If the symptom turns out to be false, then the explanation system can respond with an utterance such as

> "Well I don't believe I'm in the state of <symptom>."

On the other hand, if the symptom is true, then the SI system proceeds to identify the active cause. First it looks up the set of possible causes of this symptom (these are identified a priori). It then iterates through each possible cause, testing whether each is true. Upon knowing the actual cause, the explanation system can then respond with an utterance such as

> "I am in the state of <symptom> because I am trying to execute <active-cause>."

where <active-cause> is the name of the identified cause and in this case, corresponds to an operator node.

With this process, we assume only one node can be active among the possible causes, which means we also assume only one active path can be traced through the circuit. We hope to relax this restriction as we develop methods for constructing the abstract causal circuit.

Follow up questions

Thus far, we have described a process that generates explanations delving only one level deep in the circuit. However, the user may seek further detail in a follow up question such as "Why?" This follow up question becomes a contextual one, equivalent to saying

> "Why are you doing <symptom>?"

where <symptom> is now the name of the previously identified cause.

In order to repeat the explanation process, the SI system re-references the previously identified active cause node as the new "symptom." The SI system now repeats the same process for testing the possible causes. A subsequent explanation can then be

> "I am executing <symptom> because I believe <active-cause> is true."

As the user asks additional follow up questions, the node type of each subsequent active cause will flip-flop between proposition and operator; this reflects the nature of the abstract causal circuit. Eventually, we will explore different ways we may vary the depth—and degree of detail—to which a single explanation response is generated.

Initial implementation

Current status

We have implemented a very preliminary version of ROSEY on an indoor 2-wheel differential-drive robot (Figure 3) that attempts to explains its current locomotive behavior. ROSEY currently responds to the following queries:

- Why are you turning?
- Why are you stopped?
- Why are you reversing?
- Why are you moving?

A natural language parser separate from ROSEY handles word recognition and serves as the interface between the user and ROSEY. We borrow this parser from another system called Cerebus (Horswill, Zubek, et. al. 2000).

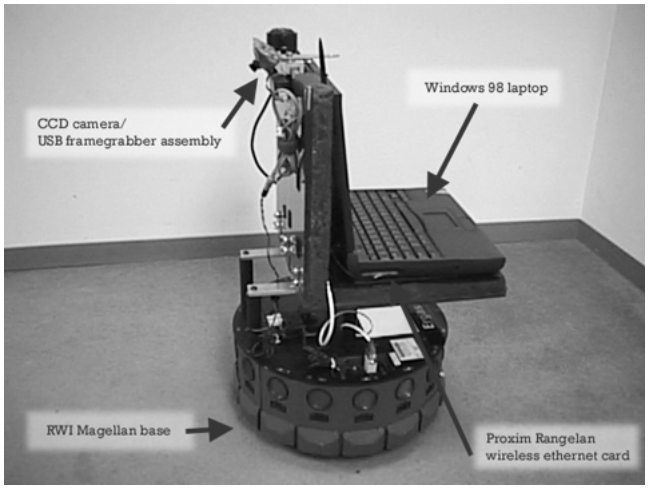

Figure 3. Indoor mobile robot platform

At the moment, ROSEY generates a one-level-deep explanation, for example:

```
> why are you turning ?
Because ROSEY is trying to get
unstuck.
```

where turning is a locomotive symptom corresponding to a proposition node. To minimize the mechanical quality of the responses, we are forgoing the use of fill-in-the-blank templates (e.g. "I am executing <symptom>", where we substitute a node name for <symptom>). Instead, every node in the abstract causal circuit has a fixed—but unique—utterance stored as an

additional property. When explaining an active cause node, the explanation system retrieves the canned response associated with this node. For instance, in the above example, the response "`Because ROSEY is trying to get unstuck.`" has been taken from the node representing the unwedger motor behavior, which is the operator currently causing `turning` to be true.

Current issues

The four propositions—`turning`, `stopped`, `reversing`, and `moving`—attempt to be descriptive of the locomotive qualities that a user may observe and ascribe to the robot as it is moving around in the world. Consequently, ROSEY recognize these words as symptoms. We want to continue to expand the set of recognizable symptoms.

To grow this set, we plan to introduce a second class of queries that ROSEY understands. So far, we have described ROSEY handling queries that pertain to its instantaneous locomotive state. We call these queries *instantaneous queries*. However, ROSEY does not handle queries that require it to maintain history or a knowledge of its past execution cycles. We call such queries *history-based queries*. This current limitation explains why ROSEY can handle an instantaneous query like "`Why are you turning?`", while failing to recognize a query like "`Why are you oscillating?`" We acknowledge that maintaining a history of states and actions will be a valuable part of a robot explanation system.

As mentioned earlier, a central issue we are facing is figuring out how to generate the abstract causal circuits. Two questions persist. First, how do we identify a sufficient set of propositions and operators? Second, what is a process for mapping circuit elements to this set? Answering the first question will be partially answered by the kinds of symptoms we see users will want to ascribe to the robot's behavior.

In addressing the second question, we are currently defining macros that permit the robot programmer to annotate the set of propositions and operators that should be reified for the explanation system. However, we plan to move towards automated techniques. In particular, we are exploring how we may leverage robot programming languages that support circuit semantics representations (Nilsson 1994). With such languages, we hope to more easily map circuit-semantic robot programs to the abstract causal circuits.

Tagged behavior-based architecture approach

To support the required runtime mechanisms, we are building ROSEY using a tagged behavior-based architecture called role-passing (Horswill 2001; Horswill 1998). In role-passing, tags are linguistic roles such as "cause" and "symptom". By using a tagged behavior-based architecture like role-passing, we gain two advantages. First, we are able to implement variable-binding, allowing us to construct higher-ordered behaviors such as the self-inspection tests. Because tags can be reassigned, the SI system is able to redirect the same computational mechanisms to different nodes in the circuit.

For instance, the SI system references the symptom node through the `symptom` tag. It then tests whether the node referenced by `symptom` is active or not. Later on, the SI system can quickly rebind the `symptom` tag to a different node and repeat the very same test.

Second, we are using role-passing in order to reuse some pre-existing infrastructure; this includes the mechanism for looking up arbitrary nodes in the circuit as well as the "active" tests performed on an individual proposition or operator.

Related systems

ROSEY is an extension of the Cerebus project (Horswill 2001; Horswill, Zubek, et. al. 2000), a system that is able to discuss its capabilities and internal structures. This work attempts to extend this ability by enabling the robot to explain how its processes and state dictate its current behavior.

ROSEY is part of a larger category of question-and-answer robots. One of those robots includes (Torrance 1994), who describes a robot that can answer questions about navigational plans and about the spatial relationships that hold between known places. KAMRO is a two-armed mobile assembly robot that can explain its own error recovery methods to a human supervisor (Längle, Lüth, Stopp, and Herzog 1996).

ROSEY is also part of a category of work that involves explanation of physical systems. A variety of work has been done, many in the context of diagnosis, instruction, and engineering design. Self-explanatory simulators (Forbus and Falkenhainer 1990) represent one large class of software that permits the user to seek causal explanations of physical processes being modeled. One system that shares similar purposes as ROSEY is described in (Gautier and Gruber 1993). Their task is to respond to behavior-related queries about a physical device by automatically generating causal explanations from a model. They attempt to avoid overly detailed explanations by applying a few heuristics for selecting salient details. We will consider similar heuristics for constructing our circuit.

Conclusion

First, ROSEY is an attempt to demonstrate how a behavior-based robot can verbalize explanations of its own behaviors. More generally, it attempts to show how a parallel-reactive architecture can support a higher-level task such as explanation. Although this work is still preliminary, we believe a behavior-based system can be just as capable as a symbolic system at accomplishing such a task. In particular, we show that behavior-based robots can obtain reflective knowledge through self-inspection.

We also view ROSEY as representative of an emerging category of *self-explanatory robots.* We are interested in increasing the reliability and cooperativeness of behavior-based robots. Although this issue has been beyond the scope of this paper, we feel that self-explanatory robots will demonstrate an important trait. By revealing their internal states and processes, such robots will be better understood and accepted by human users, whether the person is an expert robot programmer or a novice user. Eventually, as mobile robots become more commonplace, even greater interest will be placed on robots that are easy to use, easy to diagnose, and easy to maintain. Self-explanatory robots will be in service of those goals.

References

Forbus, K. and Falkenhainer, B. (1990). Self-explanatory simulations: An integration of qualitative and quantitative knowledge. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, Boston, MA., AAAI Press/MIT Press, 1990, 380-387.

Gautier, P.O. and Gruber, T.R. (1993). Generating explanations of device behavior using compositional modeling and causal ordering. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, Washington, D.C., AAAI Press/MIT Press, 1993.

Horswill, I. (2001). Tagged behavior-based architectures: Integrating cognition with embodied activity. *IEEE Intelligent Systems.* September/October 2001: 30-38.

Horswill, I., Zubek, R., et. al. (2000). The Cerebus project. *AAAI Fall Symposium on Parallel Cognition.* Cape Cod, MA, November 2000.

Horswill, I. (1998). Grounding mundane inference in perception. *Autonomous* Robots, 5: 63-77.

Längle, T., Lüth, T.C., Stopp, E., and Herzog, G. (1996). Natural language access to intelligent robots: Explaining automatic error recovery. In *Artificial Intelligence: Methodology, Systems, and Applications* (Ed. A.M. Ramsay). Amsterdam: IOS, 259-267.

Mallory, R.S., Porter, B.W., and Kuipers, B.J. (1996). Comprehending complex behavior graphs through abstraction. In *Tenth International Workshop on Qualitative Reasoning about Physical Systems*, Fallen Leaf Lake, CA, 1996, 137-146.

Nayak, P.P. (1992). Causal approximations. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, San Jose, CA., AAAI Press/MIT Press, 1992, 703-709.

Nilsson, N. (1994). Teleo-reactive programs for agent control. *Journal of Artificial Intelligence Research*, 1: 139-158.

Torrance, M. (1994). Natural communication with robots. Master's Thesis. MIT. Cambridge, MA.

Williams, B.C. (1991). Critical abstraction: Generating simplest models for causal explanation. In *Proceedings of the Fifth International Workshop on Qualitative Reasoning about Physical Systems*, 1991.