A Self-Monitoring Framework for Opaque Machines

Extended Abstract

Leilani H. Gilpin
Massachusetts Institute of Technology
Cambridge, MA
lgilpin@mit.edu

Lalana Kagal
Massachusetts Institute of Technology
Cambridge, MA
lkagal@mit.edu

ABSTRACT

Diagnostic systems for complex machines are highly specialized and cannot be applied in other domains without significant effort. Our goal is to improve the robustness of diagnostics with an adaptable monitoring framework for identifying and explaining anomalous behavior that can be easily modified for different domains or systems. We define a vocabulary for reasonable data—to precisely identify contradictions between expected and anomalous behavior and a language—to express rules, policies, and constraints/preferences of the user. We combine this framework with explanation mechanisms to describe the core reasons and support for a reasonableness judgment made by running the reasoner over the reasonable data, rules and the state of the related components.

KEYWORDS

Explainability; Knowledge Representation and Reasoning

ACM Reference Format:
Leilani H. Gilpin and Lalana Kagal. 2019. A Self-Monitoring Framework for Opaque Machines. In *Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), Montreal, Canada, May 13–17, 2019,* IFAAMAS, 3 pages.

1 INTRODUCTION

Explaining why errors occur is a two part problem. The first step is detection. Diagnostic systems have been successful in detecting errors for computers, spacecraft, and computer hackers on the Internet. However these systems are domain specific–they cannot be applied to other domains and disciplines without a sizable effort. They are also static–they are not able to be augmented. These systems need to be able to absorb feedback and learn from their mistakes to improve as error-detecting systems.

The second step towards explaining errors is to create the capability for diagnostics systems to explain. Although these systems can pinpoint what went wrong, they can rarely report *why* it happened, or *how* to fix the error in the future. Our self-monitoring system aims to answer these questions by providing an explanation of how and why errors occur.

With the need for malleable, self-explaining systems, we present a new self-monitoring framework that can impose constraints of reasonableness in multiple domains. With the input of a domain-specific knowledge base and rules, it uses a reasoner, data parser with a ontology, and an explainer to judge and explain whether the input is reasonable or not. We demonstrate how this technology can be used to detect and explain anomalies after the fact, and also motivate its use as a real-time monitoring system. We also introduce the use of explanations as a type of feedback to *learn* the rules and reasons, and make better judgments in future iterations.

Proc. of the 18th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2019), N. Agmon, M. E. Taylor, E. Elkind, M. Veloso (eds.), May 13–17, 2019, Montreal, Canada. © 2019 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

2 PREVIOUS WORK

Model-based diagnosis [6] has been extended to software faults and failures [13], reasoning about plans [2], and hypothetical reasoning [5]. Our system uses inspiration from model-based diagnostics but presents an adaptable framework to impose rules and constraints of reasonability in different domains.

Monitoring for reasonability is an open topic in computer science. Collins and Michalski present a formal ontology for reasonability [4], but it lacks a structural implementation. Another formal approach without implementation is the theory of rational action [3]. Although formal approaches are provably correct, they do not lend themselves well to an implementation. Many have tried to make ontologies and generalizations of these kinds of judgments, but they remain specific to the machine specifications [1].

Monitoring systems have also been well-studied in robotics. Livingston was a single model-based diagnostic monitoring tool that reconfigured component modes[12], but was not adaptable to other domains easily. Other work has examined how to select different planning choices through monitoring[9], while our monitoring system supplements planning choices with commonsense to make better decisions. Our work also aims to enhance explainability and trust for robotic systems, especially within robot planning[7].

Preliminary iterations of this monitoring system were domain specific[8]. The novel idea here is that we have extended that work so that the system logs and rules are in standardized formats. This means that the general monitoring framework could be used for a multitude of applications, including planning, robotic manipulation, and opaque mechanical systems.

3 IMPLEMENTATION

We demonstrate that our monitoring system can provide accurate judgments of reasonableness and convincing explanations of reasonableness by applying the system to two use cases: descriptions of perception (which could be generated from an opaque scene understanding systems), and vehicle plans (from an autonomous vehicle planning system, which could be proprietary).

3.1 Log Generation

In order for our monitor to be generic, we require that the log of the system, or data, is represented in Resource Description Framework

(RDF)[1]. RDF, a W3C standard, is a data model for interoperable and extensible data exchange. RDF was chosen so that the subsequent data, use cases, and rules could be transparently shared across different applications. For perception, we generate the log for a description by parsing for relevant concepts. For this perception description use case, we develop a set of anchor points to extract commonsense knowledge from ConceptNet[11], and primitive actions are represented as a set of conceptual dependency primitives. These conceptual dependency primitives are used to construct rules, with the actor, object, and context information as input. The details are described in previous work[8]. For vehicle planning, we extend the representation to include vehicle primitive actions like yield, move (with speed and direction), stop and turn. Context is also extended to cover external factors that are specific to vehicle planning like stop lights, pedestrians, and weather.

3.2 State and Ontology

In the case of a planning system, the log data contains two states: the previous state information, and the next intended state information. This state contains information about the vehicle state and other perceptual information that may affect vehicle state, like pedestrians or other objects in the road, and traffic light colors. In the case of a perception mechanism, there is only one state. To aggregate this state information, we extract concepts from the perception description (noun phrases, verb phrases, prepositional phrases, etc). We then search for relevant information in a commonsense database that can connect and abstract these concepts together.

To represent ontology information, we support RDF Schema[2], which allows the description of groups of related resources and the relationships between these resources. The ontology represents high level concepts that we refer to as "anchor points," which is described in previous work[8]. We have assigned these anchor point nodes to represent the broad categorizations that represent primitive acts and the constraints in our rule system.

3.3 Rules and Explanations

We require that our rules are written in AIR[3]. AIR is a Web-based rule language that is grounded in RDF and supports similar interoperability and extensibility. Its reasoner can generate and track explanations for its inferences and actions. AIR explanations are themselves in RDF and can be used for further reasoning.

AIR nicely captures the reasons and descriptions necessary to output explanations. Using python and *rdflib*[4], the output RDF file is parsed for the justifications and rule descriptions, which are combined together into a human-readable explanation. This human-readable explanation is also coupled with a report of all the rules that were fired, which could be used for feedback in future iterations of the monitoring system.

For the perception problem, we use the rules from Schanks conceptual primitives [10]. For the vehicle planning problem, we use rules derived from the Massachusetts driving manual[5]. These rules can be easily changed to express the rules of the road for other states and areas.

[1]https://www.w3.org/RDF/
[2]https://www.w3.org/TR/rdf-schema/
[3]http://dig.csail.mit.edu/2009/AIR/
[4]https://github.com/RDFLib/rdflib
[5]https://www.mass.gov/lists/drivers-manuals

4 EVALUATION

We developed our test sets based on interviews with potential customers. The perception description test set is from 100 descriptions that we previously developed for a preliminary domain-specific system[8]. The descriptions are equally split between unreasonable and reasonable, with different verbs, subjects, objects, and contexts.

For the vehicle action test cases, we developed 24 examples. These examples were generated from four lights (red, yellow, green, no light), three motions (move forward, turn, stop), and a binary choice for obstructions (pedestrian or no pedestrian). For validation purposes, we check that our monitor can determine whether a perception description or a vehicle action is reasonable or not. We labeled each description of a vehicle action or perception description with a 1 or 0 as reasonable (1) or unreasonable (0).

Our adaptable monitor system performs with 82% accuracy on the perception description data set, and it judges reasonableness with 100% accuracy on the vehicle action test set. Since there are a countable number of rules and combinations, this makes sense. However, when deploying the system in a working vehicle simulation or platform, we will need to create more sophisticated and complex rules, which may cause the monitor to perform less accurately.

To evaluate how "convincing" our explanations were, we recruited 100 users from Amazon Mechanical Turk to evaluate a set of 40 explanations, evenly split between reasonable and unreasonable judgments. The set contained 20 vehicle planning explanations, and 20 perception description examples. Participants were instructed to rate each explanation on a five point Likert scale from 1 to 5 (1 being "not convincing" and 5 being "very convincing.") The average score over all explanations was 3.94, indicating that most users were moderately convinced by the explanations.

5 CONTRIBUTIONS

In this work, we present a framework for a generic monitoring system that can *judge* and *explain* the reasonableness of an input log, given a set of rules. We chose to represent our log and state data in RDF and our rules in AIR in order to have a framework that is easy to augment, extend, and adapt to other applications.

The key idea is here is that monitoring should not be invasive. Our adaptable monitoring system was designed so that it can be attached to existing working systems to make them work slightly better; to address rare and unusual anomalies that may not be well-represented in training data.

Complex machines work fairly well in practice, but when they fail, diagnosing the root-cause is difficult. More so, developing an explanation of how and why they failed is even harder. Self-monitoring constructs, like the one proposed in this paper, are a small step towards developing more trustworthy machines that perform within the constraints of reasonableness.

REFERENCES

[1] Jose Vicente Abellan-Nebot and Fernando Romero Subirón. 2010. A review of machining monitoring systems based on artificial intelligence process models. *The International Journal of Advanced Manufacturing Technology* 47, 1-4 (2010), 237–257.

[2] James Allen, Henry Kautz, Richard Pelavin, and Josh Tenenberg. 2014. *Reasoning about plans.* Morgan Kaufmann.

[3] Philip R Cohen and Hector J Levesque. 1988. *Rational interaction as the basis for communication.* Technical Report. SRI International.

[4] Allan Collins and Ryszard Michalski. 1989. The logic of plausible reasoning: A core theory. *Cognitive science* 13, 1 (1989), 1–49.

[5] Johan de Kleer, Matthew Klenk, and Alexander Feldman. 2018. Diagnosing Alternative Facts. In *28th International Workshop on Principles of Diagnosis (DX'17) (Kalpa Publications in Computing)*, Marina Zanella, Ingo Pill, and Alessandro Cimatti (Eds.), Vol. 4. EasyChair, 159–168. https://doi.org/10.29007/fkwg

[6] Johan De Kleer and Brian C Williams. 1987. Diagnosing multiple faults. *Artificial intelligence* 32, 1 (1987), 97–130.

[7] Maria Fox, Derek Long, and Daniele Magazzeni. 2017. Explainable planning. *arXiv preprint arXiv:1709.10256* (2017).

[8] Leilani H. Gilpin, Jamie C. Macbeth, and Evelyn Florentine. 2018. Monitoring Scene Understanders with Conceptual Primitive Decomposition and Commonsense Knowledge. *Advances in Cognitive Systems* 6 (2018).

[9] Phil Kim, Brian C Williams, and Mark Abramson. [n. d.]. Executing reactive, model-based programs through graph-based temporal planning.

[10] Roger C Schank. 1972. Conceptual Dependency: A Theory of Natural Language Understanding. *Cognitive Psychology* 3, 4 (1972), 552–631.

[11] Robert Speer and Catherine Havasi. 2013. ConceptNet 5: A large semantic network for relational knowledge. In *The PeopleâĂŹs Web Meets NLP*. Springer, New York, 161–176.

[12] Brian C Williams and P Pandurang Nayak. 1996. A model-based approach to reactive self-configuring systems. In *Proceedings of the national conference on artificial intelligence*. 971–978.

[13] W Eric Wong, Ruizhi Gao, Yihao Li, Rui Abreu, and Franz Wotawa. 2016. A survey on software fault localization. *IEEE Transactions on Software Engineering* 42, 8 (2016), 707–740.