System-wide Monitoring for Anomaly Detection

Leilani H. Gilpin LGILPIN@MIT.EDU
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 77 Massachusetts Avenue, Cambridge, MA, 02139 USA

Abstract

Complex systems are unable to reason about conflicting information and behaviors. This is important for autonomous systems, where *multiple* pieces of information are utilized in critical, system-level decision-making. But existing approaches are not equipped to detect and mitigate *inconsistencies amongst parts*; each part speaks a different language, and reconciling conflicts is left to uninterpretable confidence scores. I have developed a system-level error detection architecture that applies monitoring and symbolic reasoning to full-system design. I discuss results on a simulated autonomous vehicle failure, and conclude with other potential applications.

1. Introduction

Complex systems[1] can recognize objects and trends (Zhao et al., 2019), but they cannot reason. This is especially troublesome for self-driving vehicles, where adversarial attacks (Eykholt et al., 2017) can trick a vision system into perceiving a stop sign as a 45 mph sign with a few pieces of tape, or biased algorithms become critical when they are applied to pedestrian detection (Wilson et al., 2019). These examples highlight the brittleness of opaque algorithms. But when these opaque results are input to critical decision-making, such as autonomous driving, these mistakes have *consequences*.

To mitigate these failures, I built a system-wide architecture that facilitates symbolic communication and reasoning amongst parts. I have built upon my previous work in reasonableness monitoring (Gilpin et al., 2018), and applied the methodology to *full-system design*. The internal communication is a symbolic explanation. When inconsistencies arise, an explanation *synthesizer* examines the underlying reasons supporting each explanation. The synthesizer uses a priority hierarchy to ensure high-level priorities, such as passenger safety, are met.

2. System Assumptions

Consider a complex system, which is composed out of multiple subsystems. There are two ways in which this complex system can fail. The failure or inconsistency can be local to one or more subsystems that are not doing their intended jobs, or at least one of the cooperations between subsystems failed. My key hypothesis is that when subsystems *communicate* and *argue* amongst themselves,

1. I define a *complex system* as any system with multiple parts. It could be purely computational (e.g. a complex algorithm like a deep neural network) or mechanical (e.g. a mechanical watch) or a hybrid system (e.g. an autonomous machine that is built out of mechanical systems and complex software.)

then complex systems can *more robustly* self-diagnose: the complex system can dynamically decide which subsystems to trust or discount when inconsistencies arise. Further, this diagnosis is supported with reasons: an explanation of why.

In this paper, I demonstrate how to generate and use symbolic reasons to detect and reconcile internal, conflicting behavior. I demonstrate preliminary results on simulated vehicle scenarios to motivate further work towards *internal reasoning* for anomaly and error detection.

3. Method

My method, anomaly detection through explanations (ADE) is composed of a hierarchical architecture (composed of monitored subsystems) and an explanation synthesizer to reconcile inconsistencies between subsystems.

The architecture is manually constructed with reasonableness monitors (Gilpin et al., 2018) around each underlying subsystem. Each reasonableness monitor requires subsystem outputs, contextual information, and commonsense data and rules for the *domain*. In this work, reasonableness is defined as abiding by a set of commonsense rules. For example, perceiving a person crossing the street is reasonable, but perceiving a street light crossing the street is unreasonable because it violates the commonsense rule that heavy inanimate objects cannot move. Reasonableness is a binary value, since this work is focused on detecting and explaining failures. Reasonableness scoring may be explored in future work. The outputs of the reasonableness monitors are input to an explanation synthesizer, which uses a priority hierarchy and associated rules to reconcile inconsistencies amongst subsystems.

3.1 Priority Hierarchy

The explanation synthesizer uses a *priority hierarchy* that represented (in order) the key priorities in the case of inconsistencies. For a self-driving car, the priorities are related to "safety." For example, it is important to determine which objects threaten passenger (or pedestrian) safety. Most autonomous vehicles struggle with this sort of reasoning, causing consistent starting and stopping behavior that is uncomfortable[2].

As an example, the priorities used in the synthesizer for self-driving are as follows (in order): (1) Passenger Safety, (2) Passenger Perceived Safety, (3) Passenger Comfort, (4) Efficiency. These priorities are *specifically* vague, as to avoid some of the ethical questions about specific obstacles (Awad et al., 2018).

3.2 Abstract Rules

The priorities are translated into abstract rules. These rules are hand-coded, and learning these types of rules from experience may be explored in future work. For demonstration, I specifically focus on (1) Passenger Safety, which I express as the following rule: "A passenger is safe if the

2. "Herky jerky" or constant starting and stopping has lead many vehicle manufacturers to choose to ignore objects detected with low confidence scores.

Figure 1. Passenger safety rules in Prolog for successive states `S` and `T`.

```
safe(S,T,V) :- safe_transitions(S,T,V),not(threats_between(S,T)).
threats_between(S,T) :- threat(S,O); threat(T,O).
threat(S,O) :- moving(O,S,V),locatedNear(O,self,S), largeObj(O,S).
```

Figure 2. The goal tree expanded for passenger safety.

```
passenger is safe at V between s and t
    AND( OR( safe driving at V during s and t
             AND( moving V at state s
                  t succeeds s
                  moving V at state t ) )
         OR( obj is not a threat between s and t
             AND( OR( obj not a threat at s
                      obj is not moving
                      obj is not located near
                      obj is not a large object )
                  OR( obj not a threat at t
                      obj is not moving
                      obj is not located near
                      obj is not a large object ) ) ) ) )
```

vehicle proceeds at the same speed and direction, and the vehicle avoids threatening objects." In implementation, this is written in Prolog in Figure 1.

3.3 Reasoning

I generate a goal tree of all the statements that must be verified in order to satisfy each priority. The goal tree for passenger safety in the example is in Figure 2. This goal tree is generated by backward chaining: starting from a conclusion (the resultant action of each abstract goal), I find the rules that yield that conclusion.

The output is a judgement and an explanation. If there is an inconsistency, the synthesizer will indicate which subsystem(s) to trust and which subsystem(s) to discount. In the case that no inconsistencies are detected, the synthesizer will output a summary of the subsystem outputs.

4. A Motivating Example

Consider the Uber self-driving vehicle accident scenario[3] approximately 6 seconds before impact. The key facts are that the vision system was oscillating between the labels of a vehicle, bike, and

3. On March 18, 2018, the first reported self-driving pedestrian fatality occurred. The vehicle was an Uber Technologies, Inc. test vehicle: a modified 2017 Volvo XC90 with a self-driving system in computer control mode. At approximately 9:58 p.m that evening, the vehicle struck a pedestrian on northbound Mill Avenue, in Tempe, Mari-

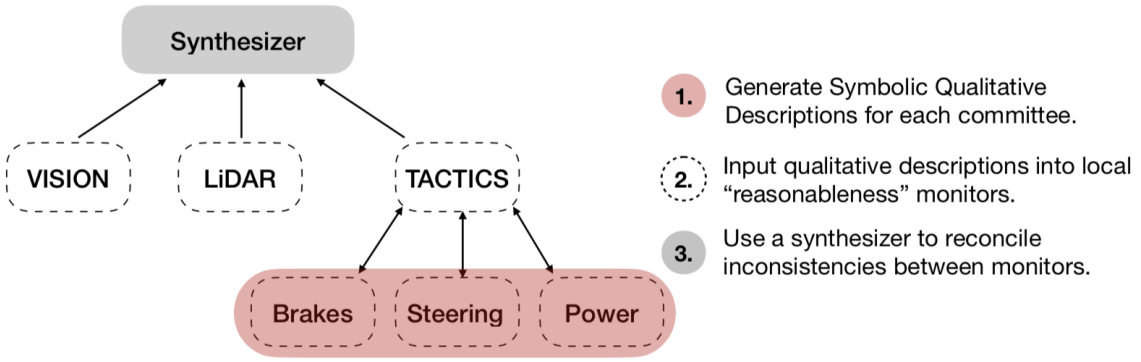
unknown object where the true pedestrian was located, the LIDAR data had detected a large moving object, and the vehicle was traveling quickly. For purpose of example, I modeled three subsystems, as seen in Figure 3.

1. Vision: the vehicle's perceptual subsystem that processes image data from an optical camera.
2. LiDAR: an active remote sensing system. The LiDAR unit emits infrared light and measures how long each light beam takes to come back after hitting a nearby object. The LiDAR unit does this millions of times a second to create a "point cloud": a 3-D map of the LiDAR's perceived world.
3. Tactics: the low-level subsystems that process operator feedback and engage mechanical components like the brakes, steering, and power controls.

Note that this is a simplified list of the subsystems in an autonomous vehicle. They are the smallest subset of subsystems necessary to reconcile the example.

Since accident data has not been released, this scenario was simulated in Carla (Dosovitskiy et al., 2017), an autonomous driving simulation platform. Logs were generated after-the-fact, and the simulated accident was repeated three times to ensure consistent results. The ADE reasoning process proceeds in three steps, as indicated in Figure 3. First, the simulated data is translated into a qualitative summary. The methods for generating these symbolic summaries was developed in prior work (Gilpin & Yuan, 2017). Note that the output of the vision system are symbolic labels. So the vision system produces a qualitative summary by default.

Figure 3. The system-wide monitoring architecture for the autonomous vehicle failure scenario. Reasoning is performed in three steps. First, qualitative descriptions are generated for each subsystem or committee. Then the qualitative descriptions are input into a "reasonableness monitor" for more symbolic support. Finally, a synthesizer checks that the monitored outputs do not contradict each other.



copa County, Arizona. Although the test vehicle had a human safety driver, they were not paying attention at the moments before impact.

4.1 Reasonableness Monitoring

The reasons are generated by monitoring the behavior of each subsystem for reasonableness (Gilpin et al., 2018). A subset of the reasonableness monitoring outputs are summarized below. For demonstration, the explanations are automatically translated into natural language. The explanations are represented in symbolic triples.

1. VISION: "This vision perception is unreasonable. There is no commonsense data supporting the similarity between a vehicle, bike and unknown object except that they can be located at the same location. This subsystem's output should be discounted."

2. LIDAR: "This LiDAR perception is reasonable. An object moving of this size is a large moving object."

3. TACTICS: "This system state is reasonable given that the vehicle has been moving quickly and proceeding straight for the last 10 second history."

4.2 Synthesizing

These symbolic reasons are passed to the synthesizer. The synthesizer finds that the LiDAR's *reason* supporting the large moving object violates an abstract goal of the priority hierarchy: `NOT ('threatening objects')`. Therefore, the synthesizer decides that the LiDAR subsystem should be trusted, and the results from the vision system should be discounted. The final output demonstrated in a natural language explanation that summarizes the synthesizer's judgement: "The best option is to veer and slow down. The vehicle is traveling too fast to suddenly stop. The vision system is inconsistent, but the LiDAR system has provided a reasonable and strong claim to avoid the object moving across the street."

5. Limitations

There may be more than one plausible, reasonable behavior with a supporting, convincing explanation. It could be difficult to decide which behavior to accept. One way to tackle this situation is to simulate alternative futures, where the simulation is able to model the behavioral and physical consequences of acting on any set of premises that may be chosen by committee arbitration. Simulating is also important for resolving conflicts between parts. If there is no direct way to reconcile a conflict, then simulation can guide the search process by finding a set of alternative premises that results in the "least bad" consequences. For each set of premises that the system might accept, simulation can expose the expected consequences. Each scenario can then be examined for dangerous conditions (for instance, collision with objects of unknown mass), and beliefs leading to those conditions lose support. It is impossible to consider every possibility, but I will explore the scenarios that are supported most readily by the available explanations.

6. Related Work

The main goal of this work is to use *internal subsystem reasoning* for anomaly detection. Other approaches detect anomalies by creating scores, such as an "interestingness" score (Geng & Hamilton, 2006). Logic-based techniques can find anomalies that do not satisfy some predicate for *typical behavior*, but in real-world environments defining "typical" behavior may not be precise. Therefore less precise, approximate reasoning like fuzzy logic (Cateni et al., 2007) allows inference on less well-defined topics.

Within the planning community, the significant parts of a complex system are *monitored* for expected behaviors. Causal-link monitoring consistently checks rules and preconditions once the desired output is established for each step, where these causal links can be qualitative (De Kleer & Brown, 1986) and input back into qualitative reasoning systems. If the monitoring system is examining actions, then action monitoring can verify preconditions right before specific actions are executed, this is typically done in RosPlan (Cashmore et al., 2015). If an error is detected, a complex system may want to repair itself. These types of self-configuring (Williams & Nayak, 1996b) systems have a flexible representation that upholds consistent design choices and principles. A self-configuring system may focus on reconfiguring components modes (Williams & Nayak, 1996a), or have a meta-process that selects between program choices (Kim et al., 2001). The MIDCA cognitive architecture is a similar approach for anomaly detection (Cox et al., 2012), but it does not handle the types of multi-modal data in autonomous driving.

Using priorities for decision-making helps in ex post facto analysis. The analytic hierarchy process or AHP (Saaty, 1988) is a method to quantify the possibilities leading up to a decision. Instead, my approach uses a similar structure but instead of quantifying possibilities, it examines the strength of the plausible options. The priority hierarchy ensures that a specific set of needs are met, and it can similarly enforce *ethical* decision making. In the self-driving car domain, this is well studied in the moral machine experiment (Awad et al., 2018). Representing these ethical dilemmas (or even, learning them) is left to future work.

7. Contributions

I contribute a system-wide monitoring method that utilizes *dynamic* explanations with a logic system to reason *between* the subsystems. This allows better system-wide communication, even with limited information, uninterpretable information, or receiving conflicting information from the underlying subsystems. I have motivated this use case for safety-critical autonomous decision making. But this is also important for enabling communication between machines and amongst machines with people; I have shown that explanations facilitate a common language for system debugging.

This paper motivates a new view of anomaly detection. A view in which failure and errors are not necessarily outliers, but inexplicable instances. This is grounded in the argument that a failure or an anomaly occurs when an explanation is inadequate or inappropriate; thus, indicating that the underlying subsystem or process should be corrected or disabled. I have shown the applicability of this view in a critical scenario like self-driving vehicle accidents, but I will apply my architecture to other applications, like argumentation and negotiation, where communication is necessary for success.

References

Awad, E., Dsouza, S., Kim, R., Schulz, J., Henrich, J., Shariff, A., Bonnefon, J.-F., & Rahwan, I. (2018). The moral machine experiment. *Nature*, *563*, 59–64.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., & Carreras, M. (2015). Rosplan: Planning in the robot operating system. *Twenty-Fifth International Conference on Automated Planning and Scheduling*.

Cateni, S., Colla, V., & Vannucci, M. (2007). A fuzzy logic-based method for outliers detection. *Artificial Intelligence and Applications* (pp. 605–610).

Cox, M. T., Oates, T., Paisner, M., & Perlis, D. (2012). Noting anomalies in streams of symbolic predicates using a-distance.

De Kleer, J., & Brown, J. S. (1986). Theories of causal ordering. *Artificial intelligence*, *29*, 33–61.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., & Koltun, V. (2017). CARLA: An open urban driving simulator. *Proceedings of the 1st Annual Conference on Robot Learning* (pp. 1–16).

Eykholt, K., Evtimov, I., Fernandes, E., Li, B., Rahmati, A., Xiao, C., Prakash, A., Kohno, T., & Song, D. (2017). Robust physical-world attacks on deep learning models. *arXiv preprint arXiv:1707.08945*.

Geng, L., & Hamilton, H. J. (2006). Interestingness measures for data mining: A survey. *ACM Computing Surveys (CSUR)*, *38*, 9–es.

Gilpin, L. H., Macbeth, J. C., & Florentine, E. (2018). Monitoring scene understanders with conceptual primitive decomposition and commonsense knowledge. *Advances in Cognitive Systems*, *6*.

Gilpin, L. H., & Yuan, B. Z. (2017). Getting up to speed on vehicle intelligence. *AAAI Spring Symposium Series*.

Kim, P., Williams, B. C., & Abramson, M. (2001). Executing reactive, model-based programs through graph-based temporal planning.

Saaty, T. L. (1988). What is the analytic hierarchy process? In *Mathematical models for decision support*, 109–121. Springer.

Williams, B. C., & Nayak, P. P. (1996a). Immobile robots ai in the new millennium. *AI magazine*, *17*, 16–16.

Williams, B. C., & Nayak, P. P. (1996b). A model-based approach to reactive self-configuring systems. *Proceedings of the national conference on artificial intelligence* (pp. 971–978).

Wilson, B., Hoffman, J., & Morgenstern, J. (2019). Predictive inequity in object detection. *arXiv preprint arXiv:1902.11097*.

Zhao, Z.-Q., Zheng, P., Xu, S.-t., & Wu, X. (2019). Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, *30*, 3212–3232.