Semi-Automated Synthesis of Driving Rules

Diego Ortiz
University of California, Santa Cruz
dortizba@ucsc.edu

Leilani Gilpin
University of California, Santa Cruz
lgilpin@ucsc.edu

Alvaro A. Cardenas
University of California, Santa Cruz
alacarde@ucsc.edu

***Abstract*—Autonomous vehicles must operate in a complex environment with various social norms and expectations. While most of the work on securing autonomous vehicles has focused on safety, we argue that we also need to monitor for deviations from various societal "common sense" rules to identify attacks against autonomous systems. In this paper, we provide a first approach to encoding and understanding these common-sense driving behaviors by semi-automatically extracting rules from driving manuals. We encode our driving rules in a formal specification and make our rules available online for other researchers.**

I. INTRODUCTION

Autonomous vehicles operate in a safety-critical, complex, open-world environment. Human drivers can abstract and apply common sense rules in this ever-changing environment, from avoiding traffic accidents to driving according to societal expectations. For example, even if you see a green light, you should not proceed if your action will result in a blocked intersection. These rules tend to be well-understood by human drivers. In fact, these rules are documented in the form of "rule books" or driving handbooks, which prepare new drivers for driving tests in different states, territories, and countries.

Furthermore, autonomous vehicles might want to behave differently depending on the region the vehicle operates. For example, some regions might allow a turn-right driving maneuver at an intersection with a red light, while others might prohibit this. Therefore, the types of driving rules are location dependent, and in fact, each state has its own "driving handbook" or "rule book". In this work, we develop a framework to extract driving rules from driving manuals in an effort to understand these rules better and to begin a discussion of whether certain rules need to be monitored to detect failures or attacks in autonomous vehicles.

In particular, the rule set can be used to *monitor* existing autonomous driving simulations for abiding by safety rules, laws, and best practices. For example, if a self-driving car is trained to drive in California, our system can monitor that the self-driving car abides by California-specific rules. The location of the vehicle can also change the prior beliefs of any autonomous agent. For example, drivers in Arizona have to be aware of dust storms–e.g., the Arizona handbook has the rule, "if you encounter a severe dust storm, reduce your speed immediately." We also find that some of the rules in driving manuals are recommendations due to different conditions of the car; for example, the rule "if you are pulling a trailer, wind currents can cause your vehicle to jackknife." These rules can be used to adjust a prior belief of any autonomous driving vehicle so it drives more cautiously. In the event of an unexplained anomaly (e.g., a jackknife), the anomaly detection agent can check if there is an explanation for it (the car is pulling a trailer) or not (a failure or potential attack). Finally, we find that certain driving rules are also dependent on additional equipment in the vehicle; for example, "if you have a transponder as you approach a toll plaza, look for and follow signs with the purple logo of E-ZPass."

To better understand these rules, we present an automated rule extraction framework that encodes driving rules in a formal specification that allows for inferences, portability, and adaptability. Our system automatically extracts 67 rules, and we manually refine 205 rules to create a corpus of 278 driving rules. As part of the contributions of this paper, we make these extracted rules publicly available online[1]. We also perform *meta analyses* with the rule set to cluster driving rules according to different types of desired features.

II. RELATED WORK

Our rule extraction system extracts driving rules from natural language text. Language explanations have been shown to help in classification tasks [9] by allowing annotators to provide supervision to a classifier via natural language explanations. Previous work on learning based on explanations [11] was demonstrated in two planning domains. While this work focuses on updating environment models, our contribution is to extract rules directly in a real-world environment. To the best of our best knowledge, this is a new contribution of using NLP and rule-based systems to learn new rules in the context of autonomous driving.

Our rule extraction system relies on a set of rules and a representation of primitive actions in autonomous vehicles. These primitive actions are outlined in prior work [7]. We extend this work by automatically extracting and analyzing safety rules from existing driving manuals. Our work is also related to the concept of "rule books" for formalizing AV behavior as a hierarchy of specifications [2]. While our goal to generate sets of rules is similar, our main objective in this

Symposium on Vehicles Security and Privacy (VehicleSec) 2023
27 February 2023, San Diego, CA, USA
ISBN 1-891562-88-6
https://dx.doi.org/10.14722/vehiclesec.2023.23083
www.ndss-symposium.org

[1]https://github.com/RollingBeatle/rule-analyzer/tree/main/results

paper is to facilitate the extraction of these driving rules from the driving authority in each region and to make them publicly available, as they represent the best practices each state has identified over several decades.

Our rule learning system is a complement to existing interlock systems [5] or formal logic [6], [14]. The key difference is that our rule language is interpretable and abstract. Our rules are represented in a simplified language (symbolic triples), and the language is abstract, using qualitative descriptions in lieu of numerical constraints. Our extracted rules can be used in these systems, and we will explore this direction in future work.

The format and representation of our extracted rules are inspired by Web standard rules and written as an object-oriented Python class. The `IF(?x), THEN(?y)` format is similar to the abstract syntax of Rule Interchange Format (RIF) [12], which is a W3C standard. Initial iterations of the rule-learning system [8] were based on Prolog and Python syntax for Prolog[2], but these languages are not adaptable and require a Prolog-style reasoner. Therefore, we wrote our new rule language in Python so that we could utilize our own NLP parsing. Other suitable rule formats include constraint logic programming languages like Datalog [3]. However, this is most applicable to database systems, which is outside of the scope of our work.

Finally, one goal of this work is to strengthen anomaly detection in autonomous vehicles by adding common sense rules. Anomaly detection is a well-studied field in the realm of machine learning [4], although learning from those anomalies is an open area. Learning from errors is promising for transition repair [10], but this is specific to the domain of robotics. Other work has strengthened anomaly detection for overcoming blind spots in autonomous vehicles [13], but this approach relies on a human in the loop, where control can be transferred to a human operator if the autonomous operator needs help. Our goal is to create the first step towards creating a set of rules that can later be used (in future work) as a specification that we can then monitor with robust anomaly detection.

III. Proposed Framework

We propose a semi-automated methodology to extract and encode driving rules from the driving handbooks from different regions. We analyze the content of the manuals using natural language processing to automate the extraction of rules and then encode them in an abstract syntax for first-order logic. We then manually check the rules and refine a subset to provide a better interpretation of them. See Figure 1 for a summary of our pipeline.

IV. Data Gathering

A. Data Collection

We look at the driving manuals of the 15th most populated states in the U.S., which correlate to the states with the most active drivers. In addition, to obtain a different perspective on driving rules, we also select the six states of Australia.

TABLE I
Pages processed for each manual.

Manual name	start page	end page	total # of pages
California	33	102	69
Texas	29	63	34
Florida	34	64	30
New York	27	70	43
Pennsylvania	40	70	30
Illinois	30	90	60
Ohio	37	70	33
New Jersey	64	157	93
Virginia	7	27	20
Washington	40	116	76
Arizona	27	70	37
Massachusetts	82	124	42
Queensland	64	153	89
West Australia	49	96	47
Tasmania	13	74	61

We collect the driving manuals by downloading them from each state's department of motor vehicles website. Each driver handbook is downloaded locally as a PDF file. We collected our dataset between June 20th and July 24th 2022. Two representative driving manuals (from Massachusetts and California) are available in our GitHub repository[3] so other researchers can test our tools.

B. Data Filtering

To analyze the PDF files, we first transform the file into a text format. We use PyPDF2[4], a free and open-source Python library that can retrieve text from PDF files.

PyPDF2 worked in most of our manuals, but we found that some of them were formatted so that PyPDF2 didn't parse correctly. We found three sources of error: (1) Orientation (when the driving manuals are in landscape mode), (2) Protection (the driving manuals prevent text scraping), and (3) Images and diagrams (when the driving manual is mostly described in diagrams and flow charts). As a result, we could not process the handbooks of three U.S. states (Georgia, North Carolina, and Michigan) and three Australian states (New South Wales, Victoria, and Northern Australia).

Furthermore, driving manuals consider a wide range of topics, such as the process for obtaining a driving license or regulations for responding to traffic infraction tickets. Since this information is not related to driving rules, we exclude several pages of the driving handbooks from our tools. In particular, we focus on the specific chapters or sections of the driver handbook that correspond to safety. These pages are manually selected. A summary of the pages that we feed our tools for each manual is shown in Table I.

V. Rules Extraction

After selecting the specific pages of a driving manual, we then use natural language processing and a novel rule

[2]https://pypi.org/project/pyprolog/

[3]https://github.com/lgilpin/driving-rules

[4]https://pypi.org/project/PyPDF2/

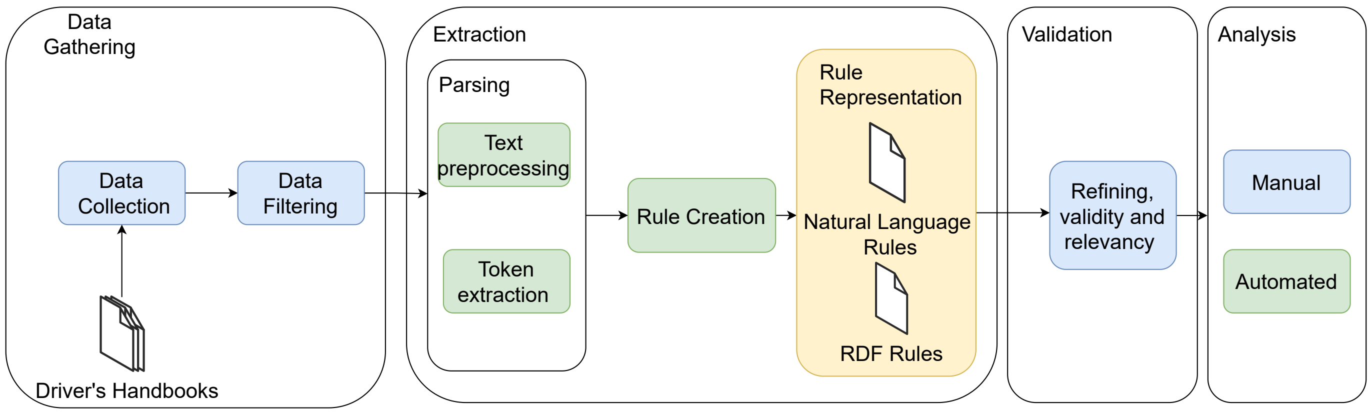

Fig. 1. Overview of our analysis methodology. We use NLP to extract rules into a formal representation, and then manually refine a subset of them.

extraction method to encode conditional propositions in an RDF triple-store format[5]. The output of the rule extraction method is a text file of rules with one rule per line.

A. Rule Format

Our rules are represented as strings in the format: `"IF(?x), THEN(?y)"`, where `(?x)` is the antecedent of the rule, and `(?y)` is the consequent of the rule. We represent the antecedent and consequent in two ways: (1) in natural language and (2) in a conjunction of a triple store representation.

This format allows the rules to be standardized, portable, and adaptable. The standardized format allows transformations, matching, and inference on the rules can be done systematically. For example, our rule list can be combined with a modus ponens inference method, and the antecedent and consequent of the rules can be matched without complicated parsing processes. Rules can also be translated into equivalent logical sentences.

Secondly, the portability of the rules allows a systematic encoding of driving safety expertise. This expertise is programming language agnostic: the rule list is usable whether the underlying system is in Python, C, JavaScript, etc. The rules do not have to be translated into different languages. And finally, the rules are adaptable; they can be altered, changed, and removed without affecting the underlying system. Whereas, if the rules were written in a software system, there would be dependencies between rules that would be to be altered.

B. Natural Language Processing

Rules are created by parsing the natural language text. Our system parses each page into a list of strings. Each string is one sentence. Then, the sentence is parsed into word tokens. If any of the rule keywords: `IF, THEN, BECAUSE` are in the sentence, then a rule is flagged to be made.

If any keywords are found in a sentence, the sentence is flagged as a potential rule, and we proceed to the next step of our analysis. Our rationale for using these keywords is the following:

- Syntax: Since we are creating if-then rules, the natural language rule description must contain an antecedent (an if clause) and a consequent (a then clause). If either the antecedent or the consequent is not found, then the rule extraction fails for the input.
- Soundness: Conditional propositions may be encoded in different natural language expressions, but looking for these might produce several false rules. Because our goal in this work is to get an initial sense of the diversity of driving rules, we focus on keywords that will produce If-Then rules with high specificity, even if we miss some.

We split each identified sentence into the *if-phrase* and the *then-phrase* corresponding to the antecedent and the consequent of the rule. First, the sentence is split by non-alphanumeric characters: `:,.` into a list of clauses. Then the *if-phrase* and *then-phrase* are set with the following rule-based method:

- If there are exactly two clauses, return the if-clause as the first item and the then-clause as the second item if `IF` is in the first clause. Otherwise, reverse the order of the clauses.
- If there is one clause, split the clause on `IF` and set the if-clause to the first part of the split and the then-clause to the last part of the split.
- If there are more than two clauses, then the if-clause is the first clause, and the then clause is the composition of the rest of the clauses.

After the *if-phrase* and the *then-phrase* are set, then the natural language representation is completed. The rule is returned as `IF(if-phrase), THEN(then-phrase)`. If the triple-store representation is flagged, then the `make_triples_from_phrase` function is called. This function makes a set of triples for the `if-phrase` and the `then-phrase`. The `make_triples_from_phrase` function receives an input phrase: a natural language phrase corresponding to the `if-phrase` or the `then-phrase`. The

[5]RDF: https://www.w3.org/RDF/

function searches the input phrase for a conjunction keyword: `and, that, not` or `or`.

The `make_triples_from_phrase` function is recursively called until no conjunction keywords are found. The function searches for conjunction keywords: `and, that, not` or `or` and performs the following:

1) If a conjunction keyword "and" or "that" is found in the input phrase, then an AND phrase is created: `AND(?x, ?y)`
2) If instead "or" is found in the phrase, then an OR phrase is created: `OR(?x, ?y)`
3) Otherwise if `NOT` or `NEVER` is found in the phrase, then a NOT phrase is created: `NOT(?x)`

The `(?x)` and `(?y)` parameters are the two phrases of the conjunction. For example, in the sentence: "I yield at the stop sign and I continue through the intersection," the two phrases are "I yield at the stop sign" and "I continue through the intersection." In the `make_triples_from_phrase` function, if a complex conjunction, e.g., AND or OR is found in the input phrase, then the phrases are split on the conjunction keyword. In the NOT case, the negation keyword is removed from the phrase. At this point, each phrase is converted into a set of triples.

C. Triple-Store Rule Creation

Each of the `(?x)` is a list or singleton of RDF-style triples. The triples are made by using a part of speech (POS) tagger from the NTLK[6] library, especially `nltk.tokenize.word_tokenize`.

1) Generating Triples: Triples are generated by using a part-of-speech (POS) tagger and extracting the subject, verb, and object. The nltk POS tagger returns the list of tokens as a list of tuples of tokens, where each token is associated with a POS tag indicating whether the word is a noun, verb, adverb, adjective, etc.[7]. In creating triples, we proceed in the following steps to make a triple consisting of a `(subject, verb, object)`:

1) Set the verb by finding the first POS tag that starts with a "V". The verb is set by setting the verb. There are list of "special verbs," e.g., to be and to have which are represented with `IsA` and `HasA` which are semantic relations used in the semantic web, knowledge base, and symbolic AI community [1].
2) Set the subject by finding the first POS tag that starts with a "N" before the verb. If no subject is found, then the keyword `self` is used. This assigns the subject of the sentence to the ego vehicle.
3) Set the object by finding the first POS tag that starts with "N" after the verb.

VI. Validation

We validate the rules manually. Overall, we obtained 67 rules that did not need any manual refinement at all. The following example shows one of the rules extracted by our method, informing drivers that if they go through an intersection but cannot advance (i.e., if they will block the intersection), they may get a ticket:

`IF (self, block, intersection), THEN (self, get, ticket)`

As explained before, our automated tool can extract rules that contain logical operators such as `AND OR` and `NOT`. For example, the following rule tells the driver that if they see a train, they should wait until it passes and then cross the tracks:

`IF (self, see, train), THEN AND((wait, until, passes), (self, cross, tracks))`

A. Refinement

Some of our automatically extracted rules require modifications before they can be used. In this initial prototype, our parsing tools are not perfect, and thus we need a step to refine some of the extracted rules. For example, the following automatically-extracted rule is incomplete:

`IF (Dont, isA, ashing), THEN (Drivers, yield, pedestrians)`

This rule has two problems, first, the subject is incomplete (it should be the `Don't walk` sign), and second, our parser had problems with some words (when we looked at the extracted sentence, the original sentence had the word `flashing`, but our PDF parser translated it as `fl ashing`). Therefore, we manually refined the extracted rule to the following proposition.

`IF (Don't walk, isA, flashing), THEN (Drivers, yield, pedestrians)`

To illustrate another example of the types of rules that needed manual refinements, consider the following rule:

`IF (self, isA, intersection), THEN (continue, through, intersection)`

While the above rule appears to be semantically correct, when we looked at the original sentence from which this rule was created, we discovered it was missing the `AND` logical operator in the antecedent. The sentence is "if you are in an intersection when you see an emergency vehicle, continue through the intersection." We, therefore, refined the rule as:

`IF AND((self, isA, intersection),(self, see, emergency vehicle)) THEN (continue, through, intersection)`

Overall we used 205 automatically-extracted rules as a starting point and refined them to create valid rules. We emphasize that we did not read the manuals, nor created the rules completely manually. We only looked at the extracted sentence from our parser (sentences satisfying if-then conditions), and the associated rule created from that sentence. If the semantics of both statements did not match, we refined the extracted rule to match the intended semantics of the original sentence.

B. Relevancy

Finally, while we focused on parsing the relevant parts of the driving manuals, we still obtained several rules that are not directly related to driving. We still keep these extracted

[6]nltk library: https://www.nltk.org/

[7]The full list of POS tags are available here: https://www.nltk.org/book/ch05.html

rules because they may be used to expand the functionality of autonomous driving agents. For example, an autonomous agent might inform the human driver that it can place a call to an emergency line: "if a dangerous condition exists at a rail crossing call the number listed on the emergency sign"

`IF (dangerous condition, at, rail), THEN (self, call, emergency number)`

Autonomous agents can also remind human drivers of potential penalties:

`IF (self, isA, hit-and-run), THEN (punishment, isA, severe)`

Finally, some of these non-driving rules can be helpful for autonomous driving agents to inform their prior beliefs on how other objects near the car may behave. For example, we extracted rules regarding motorized bicycles, such as the following rule telling motorized bicyclists to use caution with others,

`IF (self, drive, motorized bicycle), THEN (use caution, avoid, other bicyclists)`

VII. Results

Our approach automatically identified 67 semantically-correct rules, with an extra 205 rules obtained by manual refinement. We also identified 246 extraneous rules not directly related to driving actions (e.g., warnings about financial penalties, or calling specific phone numbers). We summarize these results in Table II. The results are compiled in a CSV file available in our GitHub repository[8].

TABLE II
Total number of rules extracted automatically, refined, or discarded.

State/Territory	Automated	Refined	Extraneous
California	8	31	27
Texas	12	15	4
Florida	0	15	9
New York	10	20	14
Pennsylvania	9	19	15
Ohio	6	6	6
Illinois	2	18	22
New Jersey	3	17	24
Virginia	0	11	16
Washington	3	9	3
Arizona	3	15	21
Massachusetts	6	13	58
Queensland	7	18	19
Tasmania	0	0	2
West Australia	2	6	4
Total	67	212	245

A. Least Number of Driving Rules

As we can see in the table, we were able to extract several rules in some states–e.g., 38 driving rules (automated and refined) in California or 30 in New York–while in some other regions, we got very few rules–e.g., Tasmania with 0, West Australia with 8, Ohio with 12 and Washington with 12. In trying to understand the reasons for these discrepancies, we found that they originated from three main problems:

1) PyPDF2 couldn't parse properly the file and produced broken words.
2) Some manuals make heavy use of images and diagrams to express driving rules, and therefore, our parser cannot extract them.
3) The writing style of some manuals did not match the tokens we were looking for.

For example, in some manuals, PyPDF2 had issues extracting the text, as in the case of Tasmania. For example, the following rule,

`IF (travel, oo, ride), THEN (self, co, b)`

corresponds to the following extracted text warning the driver that a rider might come off their bicycle:

`if yo u travel t oo close to a ride r they co uld come off their b icyc le`

Our automated tool was unable to extract rules from this broken English. After seeing these problems with the parsing from PyPDF2, we attempted to use other tools for extracting text, but we got similar results. In future work, we will look into better parsing tools to extract text from PDF files.

With West Australia, we found that the format of the manual didn't allow PyPDF2 to find text. While the manual does have some text that can be parsed, most of the driving rules are within figure boxes, and the text in the figure boxes was not parsed by PyPDF2, as the graphics interrupted the reading process.

Finally, some driving manuals have different styles that affect our parser. For example, the driving manual of Ohio has several statements written as cases, such as "Drive on the right half of the roadway except:" followed by a list of cases when the driver should avoid this behavior as:

- When overtaking and passing another vehicle proceeding in the same direction.
- When driving on a road divided into three or more marked lanes.
- When driving on a road designed and posted with signs for one-way traffic.
- When otherwise directed by a police officer or traffic control device
- When an obstruction makes it necessary for you to drive left-of-center.

This style is not appropriately read by our parser, thus preventing us from extracting these rules.

Similarly, the driving manual of Washington does not follow the standard if-then form that our parser works on. Important information that would generate rules is posted in a similar listing format that is not easily read by our parser. As an example take this list of speeds allowed on different roads from the Washington Driver Guide:

"Obey speed limit signs. They are there for your safety. Speed limits, unless otherwise posted, are:"

- 20 mph in school zones.

[8]https://github.com/RollingBeatle/rule-analyzer/tree/main/results

- 25 mph on streets of cities and towns.
- 50 mph on county roads.
- 60 mph on state highways.
- Parts of interstate highways may be posted with higher maximum speeds.

B. Largest Number of Driving Rules

We now turn our attention to the state with the largest number of driving rules: California. This outlier result is interesting because we feed our tool more pages to analyze from the driving manuals of New Jersey and Queensland, but we end up with half of the rules compared to the number we obtained from California.

In analyzing why we got these results, we found that the writing style of the driving manual of California was particularly well-suited for our rule extraction method. Most of the rules were expressed as if-then statements in sentences after section subtitles.

In addition, New Jersey and Queensland explain different behaviors using diagrams and tables, which our parser cannot process.

C. Outlier in Extraneous Rules

Other manuals had the opposite results, as many rules were successfully extracted, with cases such as Massachusetts and California having the most extracted rules in total (including the extraneous rules) with 77 and 66, respectively.

Massachusetts had by far the largest number of extraneous rules (58) which is more than double the maximum number of rules of the next state (California with 27). We found that the reason for this is that the safety section of the Massachusetts driver's manual also contains motorcycle driving rules as well as emergency response rules.

An example of the type of rules that we extracted for motorcycles came from the following text: `if you are operating a motorcycle you may only pass single file`

This sentence, after being parsed by our extractor, yielded the following result:

```
IF (self, isA, motorcycle), THEN (self, pass, single)
```

Likewise, an example of the extracted rules for the interaction with emergency response personnel came from the following text describing an encounter with an officer:

```
if you have questions about the citation you can ask the officer to clarify
```

With the above text, our extractor produced the following rule

```
IF (questions, about, citation), THEN (self, ask, officer)
```

Finally, some of the extraneous rules might be considered driving rules, but in our manual classification, we decided to leave them under the "extraneous" label because they may require a new way in which autonomous vehicles should interact with police officers. For example, we found a couple of rules that manage encounters with police officers and state what to do when an officer directs the driver.

```
if the officer directs you to pull over in a certain place pull over where directed
```

With the previous text, the extractor generated the following rule.

```
IF (officer, in, place), THEN (pull, over, directed)
```

With enough sophistication in autonomous vehicles, this rule could be clustered under the "automated" label in Table II. However, we point out that this rule is an example of the common-sense rules that we wanted to find when we started this work: rules that show the complexity of interactions and expectations that autonomous vehicles will face, and that need to be addressed.

D. Equivalent rules

To be considered equivalent, rules generally must have the same number of triples and share the exact words or synonyms in the triple for both antecedents and consequent. However, rules with additional terms are considered equivalent if their addition does not change the meaning in either antecedent or consequent. Furthermore, we checked the extracted text to ensure that the rule equivalences also matched the meaning of the text they were extracted from.

Take the following rules extracted from the state of Florida and the state of Texas.

```
IF (vehicles, at, same time), THEN (driver, yield, right)
IF (vehicles, at, same time), THEN (yield, on, right)
```

These rules share the same antecedent. Their consequent, although different by an additional word and organization, are equivalent in their meaning. In total, we found 13 rules that had equivalency between them. These equivalences came from rules extracted from matches between states like Ohio, Florida, California, New Jersey, Pennsylvania, and Texas.

E. Region-Specific Rules

The driving manuals can also give us insights into the unique challenges that drivers from different regions might experience. For example, the following rule:

```
IF (self, encounter, duststorm), THEN (self, Reduce, speed)
```

Represents the rule extracted from the following text "If you encounter a severe dust storm, reduce your speed immediately." This rule was found in Arizona, but we didn't find similar warnings in driving manuals from other states.

Another obvious regional difference is the side of the road you drive on and the associated rules. For example, drivers in Australia must drive on the left-hand side of incoming traffic, while drivers in the United States must drive on the right-hand side of the road.

For example, the following rule indicates how a vehicle should behave in West Australia if it is moving at a lower speed than other traffic.

```
IF (self, isA, movingslow), THEN
(moveleft, allow, traffic)
```

This rule was extracted from the following text:

```
if you are driving a slow moving vehicle
pull well over to the left to allow
following trafc to overtake
```

In contrast, the following rule in California suggests the opposite behavior.

```
IF (self, choose, slow), THEN NOT(self,
do, left lane)
```

Extracted from the following text:

```
if you choose to drive slower than other
traffic do not drive in the fast lane
```

F. Classification of Rules

To facilitate the use of the rules we are making available online, we decided to group them based on their semantics. After going over our extracted driving rules, we identified the following four categories: Vehicle Behavior, Road Signs, Environmental Conditions, and Safety Hazards. We use these categories to define a scope to fit each rule and also allow us to differentiate the situations in which the rule should apply.

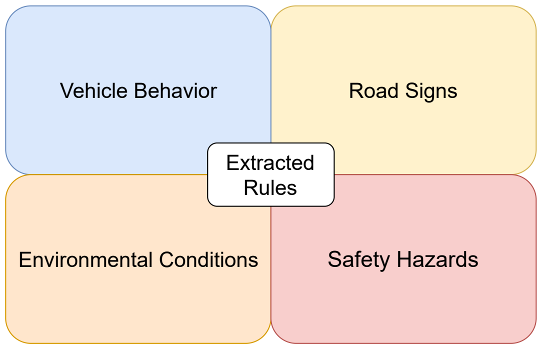

Fig. 2. Driving Rule Categories

We now give examples of some of the rules under each category.

Vehicle Behavior: General driving rules that may depend on other drivers.

- `IF AND (self, hasA, light) (vehicles, in, intersection), NOT THEN (go, across, intersection)`
- `IF (self, miss, exit), THEN (self, go, next one)`

Road Signs: Vehicle interaction with traffic signs such as the yellow lines on the road, yield signs, etc.

- `IF (arrows, indicate, directions), THEN (self, drive, directions)`
- `IF (load, greater than, figure), THEN NOT(Do, try, enter)`

Environmental Conditions: How different events, such as fog, snow, or black ice, influence how the driver should react.

- `IF (fog, becomes, thick), THEN (self, pull off, road)`
- `IF AND(day, starts, raining),(day, is, hot), THEN (pavement, isA, slippery)`

Safety Hazards: Rules in an event that may cause imminent harm to the vehicle and its passengers, such as crashing against an animal or how to handle the vehicle in case of break failure.

- `IF (vehicle, begins, losetraction), THEN (self, decrease, speed)`
- `IF (self, rear, skid left), THEN (self, turn, wheel left)`

These categories can be used as a starting place to assign priorities to driving rules. For example, when a vehicle has to follow multiple conflicting rules, it may give the highest priority to safety hazards and less to environmental conditions, which appear to be mostly recommendations.

G. Impactful Concepts

We now look at the types of words and bigrams in our extracted rules. A simple Word Cloud analysis of the rules can be seen in Figure 3. Most of the words are commands (stop, pull, yield, drive, brake, wait, etc.) To better understand the most important concepts, we look at bigrams.

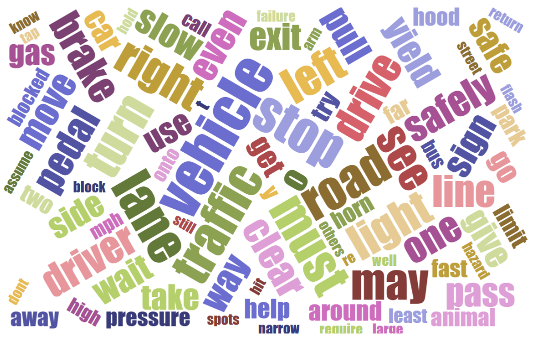

Fig. 3. Word Cloud of Extracted Driving Rules

TABLE III
SIGNIFICANCE SCORE EXTRACTED FROM RULE SET

Bigram	TF-IDF Score
self isa	11.50
vehicle self	3.32
line stop	2.72
stop intersection	2.70
self stop	2.69
intersection self	2.54
pull road	2.53
self miss	2.39
isa line	2.36
self pass	2.28

We decided to use word bigrams instead of single words. Bigrams allow us to conserve some structure of the statements declared on the rules and avoid missing statements with complex conditions that may repeat themselves through the rule set.

We decided to use Term Frequency-Inverse Document Frequency (TF-IDF) to calculate the impact of the bigrams in the rule set. TF-IDF is a technique used in natural language processing that calculates the importance of a word in a collection of documents by calculating its frequency in each document and offsetting it with the number of times its found in the documents. This helps filter common words without actual impact and get the most relevant ones assigning them higher scores. These scores help to filter the rule set of connectors and other words that may lack impact

In Table III we have collected the top 10 TF-IDF scores from the rules set showing the most significant bigrams in our set. The top score for the bigram "self isa" indicates that many of the rules we extracted describe how our vehicle "ego vehicle" should behave. It is a clear outlier, which means that a large portion of the rules depend on the condition of the ego vehicle. However, the next bigrams display specific behaviors that illustrate interactions with road signs, such as "line stop" and "stop intersection" and behaviors such as missing an exit or passing another vehicle. Their significance shows that there are common factors between all the manuals that can help to generate a comprehensive set of rules that may apply to all vehicles.

VIII. Limitations

In this initial paper, our main goal was to demonstrate a proof of concept for extracting driving rules, identify interesting common-sense rules, and share them with other researchers.

There is room for improvement in various areas. The first is to find or develop more robust text parsing tools. As explained in our section on refining rules, a large percentage of the rules that needed refinement were originally parsed incorrectly, mostly by the parser chopping words into letters separated by spaces. We focused on extracting driving rules from PDF manuals, as PDF files were available from all our target states. However, some states are releasing manuals in HTML format, and this might make parsing the text a bit easier. We will look into this option in future work.

After obtaining accurate text, the next room for improvement is to guarantee good levels of soundness and completeness for the rules generated. Ideally, we want that all extracted rules are relevant and accurate, and only those. Our tokenization looking for if-then rules guarantees that we will get a conditional statement, but the expressiveness of this rule might not capture more complex situations.

Another limitation is that our contribution is a best-effort approach. In future work, we will include a formal coverage analysis of the rules that are extracted. This will include developing a novel validation data set of driving rules.

In future work, we would like to expand our rules to include those that use temporal logic, and other types of logic: first-order predicate calculus, event calculus, etc. To use those logics, we can extend our rule keyword set to include other options, e.g., `whenever`, `as long as`, etc.

Finally, our next step will be to codify these rules into a specification that can be tested. We want to identify what are the challenges for autonomous vehicles in implementing the various common-sense rules available in driving manuals and also to see if they can be monitored for compliance. Our goal is to develop tests that can detect when autonomous vehicles violate some of these rules.

IX. Conclusions

Our goal in this paper was to extract a set of rules that show the variety of rules, conditions, and recommendations that future autonomous vehicles will have to deal with, even if partially so. We extracted overall 278 driving rules that we make publicly available to other researchers. We also extract 246 rules that provide some context on driving situations. In our future work, we plan to implement a subset of these rules in an anomaly detection system that can identify deviations from the desired specification. We will also explore extracting more complex rules defined in temporal logic, as well as the inclusion of rules as prior beliefs for the autonomous agent.

Acknowledgments

This research is supported in part by NSF CNS-1929410, NSF CNS-1931573, and by a Seed Funding Initiative from the Office of Research at UCSC.

References

[1] Ronald J. Brachman. What is-a is and isn't: An analysis of taxonomic links in semantic networks. *Computer*, 16(10):30–36, 1983.

[2] Andrea Censi, Konstantin Slutsky, Tichakorn Wongpiromsarn, Dmitry Yershov, Scott Pendleton, James Fu, and Emilio Frazzoli. Liability, ethics, and culture-aware behavior specification using rulebooks. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8536–8542, 2019.

[3] Stefano Ceri, Georg Gottlob, Letizia Tanca, et al. What you always wanted to know about datalog(and never dared to ask). *IEEE transactions on knowledge and data engineering*, 1(1):146–166, 1989.

[4] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[5] Jeff Chow, Valerie Richmond, Mike Wang, Uriel Guajardo, Daniel Jackson, Nikos Arechiga, Geoffrey Litt, Soonho Kong, and Sergio Campos. Certified control: A new safety architecture for autonomous vehicles.

[6] Daniel J. Fremont, Tommaso Dreossi, Shromona Ghosh, Xiangyu Yue, Alberto L. Sangiovanni-Vincentelli, and Sanjit A. Seshia. Scenic: a language for scenario specification and scene generation. In *Proceedings of the 40th ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)*, pages 63–78, 2019.

[7] Leilani H. Gilpin and Lalana Kagal. An adaptable self-monitoring framework for complex machines. *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2019, Montreal, Quebec, May 13-17, 2019*, page 3, 2019.

[8] Leilani Hendrina Gilpin. *Anomaly detection through explanations*. PhD thesis, Massachusetts Institute of Technology, 2020.

[9] Braden Hancock, Paroma Varma, Stephanie Wang, Martin Bringmann, Percy Liang, and Christopher Ré. Training classifiers with natural language explanations. *arXiv preprint arXiv:1805.03818*, 2018.

[10] Jarrett Holtz, Arjun Guha, and Joydeep Biswas. Interactive robot transition repair with smt. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 4905–4911. AAAI Press, 2018.

[11] Matthew Molineaux and David W Aha. Learning unknown event models. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[12] Leora Morgenstern, Chris Welty, Harold Boley, and Gary Hallmark. Rif primer. *W3C Recommendation*, 22:190, 2010.

[13] Ramya Ramakrishnan, Ece Kamar, Besmira Nushi, Debadeepta Dey, Julie Shah, and Eric Horvitz. Overcoming blind spots in the real world: Leveraging complementary abilities for joint execution. 2019.

[14] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.