Check Yourself Before You Wreck Yourself: The APEX Framework for Safe Autonomous Vehicles

Matthew O'Kelly, Houssam Abbas, Rahul Mangharam
Department of Electrical and Systems Engineering
University of Pennsylvania, Philadelphia, PA, USA
{mokelly, habbas, rahulm}@seas.upenn.edu [*]

ABSTRACT

The goal of the APEX framework is to test and verify the decision controllers of autonomous vehicles (AVs). If AVs are to be considered drivers, as the NHTSA tepidly recommended to Google in a recent letter, we must derive new solutions to bound the risk that the vehicles software poses to society.

Complete AV safety requires that the consequences of the software agents actions are considered over the set of all possible configurations. The APEX framework addresses the challenge of guaranteeing AV safety via the automated proof of of system properties expressed in first order logic. The approach begins with a scenario description language which provides a means of instantiating autonomous agents within a variety of multiobjective scenarios. To this end, we address the models used within the APEX agent library, and the structure of a scenario in the context of the hybrid systems formalism.

Given an APEX scenario description, the tool automatically generates executable hybrid models, which are proved to be safe or unsafe with respect to the specification governing the system. We consider both the use of delta-decision procedures to generate formal proofs, and enhancements to such algorithms using robustness metrics derived from guided simulation. In order to provide actionable feedback for industrial scale users, we demonstrate the concretization and visualization of counterexamples.

1. INTRODUCTION

1.1 A Driver's License for AVs

AVs are already driving on public roads and the widespread acceptance of the technology by major manufacturers seems nearly inevitable. Several well publicized incidents involving AVs have already happened; while some in academia have argued for a utilitarian view of vehicle ethics, the problem of AV safety is far more nuanced than a simple $\#define$ set to the wrong value. The APEX framework aims to create a tool which supports the analysis of both accidents that have already occurred, and to support engineers in the early phase of designs where the ramifications of discrete decisions filtered through non-linear vehicle dynamics and poorly characterized uncertainty in perception are difficult to discern.

A key element of AV safety is the understanding that fixing the problem in a single instance of a scenario does not tell you that you killed the bug. If a deployed AV exhibits unsafe behavior, how does the engineering team responsible for fixing the bug know that the patch will not cause unsafe behavior in another configuration of the scenario? The APEX framework utilizes both the pragmatic results of robustness guided testing and the heavy machinery of formal methods to provide detailed, actionable analysis for realistic models of AV decision software.

1.2 Core Assumptions

1.2.1 *Markovian Process*

Can express ontology using state space representation with a finite number of state variables.

1.2.2 *Bounded Error*

Error in sensing are lie in closed sets. We do not assume a distribution of such errors. The intent is to capture the safety of the vehicle if the specification of the system is met. Larger errors may be introduced to investigate the vehicle's behavior if the implicit or explicit contracts between sensing and planning are not met.

1.2.3 *Expressivity of Modeling Language*

The AV is modeled as a hybrid system, while the hybrid system formalism can capture most features of AV code the expression of some algorithms is hindered by the lack of data structures and traditional loop constructs. We simplify the creation of driving scenarios and vehicle models with a domain specific language which generates executables expressed as a hybrid system.

1.2.4 *Scenario Coverage*

The verification problems which APEX investigates cover only a portion of the possible executions of a AV operating in an open world. However, because there are often no guarantees about even the simplest AV scenarios with

[*]This work was supported in part by the TerraSwarm Research Center, one of six centers supported by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA.

ICCPS '17, April 18–21, 2017, Pittsburgh, Pennsylvania, USA.
© 2017 ACM. ISBN 978-1-4503-3955-1/16/04...$15.00
DOI: http://dx.doi.org/10.1145/2883817.2883841

dynamic traffic participants, each verified AV scenario is a valuable evidence based argument that the AV will perform safe actions which are difficult to express in the traditional language of control theory. Much like driver's tests cover core driving scenarios in order to bound the risk that a human driver poses to the public, we envision a large library of scenarios which all AV software awaiting deployment will be checked against before over the air updates or release.

1.3 Contributions

1.3.1 Domain Specific Language

The APEX DSL simplifies the creation of complex driving scenarios consisting of diverse agents. The language supports parallel composition of agent models and compilation to executables for simulation, falsification, and verification. The DSL encodes a bottom up engineering effort as agent models which may be instantiated from the top down in order to create diverse scenarios.

1.3.2 Execution Engine

The framework supports translation, execution, and scripting of a variety of verification tools including s-TaLiRo and dReach. The intermediate representation provided by the scenario instantiation allows new verification tools to be prototyped quickly and efficiently using existing solvers. In addition scenarios represented in APEX can be used to generate SMT2 encodings for benchmarking and distribution.

1.3.3 Scenario Library

The framework is distributed with three example scenarios. Each scenario is extensible and reconfigurable. More specifically objects such as the road, number of vehicles, traffic laws, and environmental behavior are defined parametrically. Thus, entering new scenarios is simplified.

1.3.4 Photorealistic Visualization

A key feature is the visualization of counterexamples within a photorealistic simulation. As will be discussed in section XX, counterexamples discovered in APEX may either originate from a simulation based falsification effort or verification engines such as dReach. If the counterexample is returned by a verification tool the intervals describing the existence of such an execution are explored by the falsification engine and *at least* one counterexample is returned for visualization. Often times there is a family of counterexamples within the intervals, in that case APEX can return multiple views of executions which are unsafe.

1.3.5 Notes

We want to be able to verify driving scenarios before the car is on the road, And recreate accidents to understand whether the decision controller made a 'mistake. Assumption: KNOWN (not NO) bounded perception errors, but no false negatives. Arguing for the realism of the driving scenarios that we are able to verify. Trajectory tracker Trajectory planner 7D vehicle dynamics Curved road Multiple vehicles Counterexample synthesis and export to realistic simulation environments: Unity Arguing for the ease of use of the DSL: Modular (can add and remove agents) Predefined agents / object-oriented Once scenario is described, will automatically generate files for falsification and verification (with dReach) with formal semantics Comparisons: Abstraction-based techniques conflate multiple trajectories, some of which are safe and others are not. LUT used by real trajectory planner causes many modes in the hybrid system, which presents difficulties to the reachability tool. NN gives a guaranteed approximation, and gets rid of the added modes. Test cases Straight line similar to Tesla scenario Curved highway T-junction

Base software which will enable series of papers. Key results here will be quick SAT deep in the execution and concretization of dReach counterexamples into family of executions for multiple views of bugs. Backchaining under-approximations can also work here if we expand ball at lowest robustness point. Target: ICCPS 17 (OKelly, Abbas, Mangharam). Target: SAE World Congress 17 (OKelly, Abbas, Mangharam). Quick falsification deep in scenarios with many mode switches Incremental data parallel verification Fine grained approximation of the satisfiable core(s) Export to IR to executables for simulation, robustness estimation, and verification Demonstrate counterexample in complex neural network. What guarantees can we give about complexity, soundness, and completeness of the approach. Matt: Sound and Complete.

1.4 Organization

2. THE *BODY* OF THE PAPER

Typically, the body of a paper is organized into a hierarchical structure, with numbered or unnumbered headings for sections, subsections, sub-subsections, and even smaller sections. The command `\section` that precedes this paragraph is part of such a hierarchy.[1] LaTeX handles the numbering and placement of these headings for you, when you use the appropriate heading commands around the titles of the headings. If you want a sub-subsection or smaller part to be unnumbered in your output, simply append an asterisk to the command name. Examples of both numbered and unnumbered headings will appear throughout the balance of this sample document.

Because the entire article is contained in the **document** environment, you can indicate the start of a new paragraph with a blank line in your input file; that is why this sentence forms a separate paragraph.

2.1 Type Changes and *Special* Characters

We have already seen several typeface changes in this sample. You can indicate italicized words or phrases in your text with the command `\textit`; emboldening with the command `\textbf` and typewriter-style (for instance, for computer code) with `\texttt`. But remember, you do not have to indicate typestyle changes when such changes are part of the *structural* elements of your article; for instance, the heading of this subsection will be in a sans serif[2] typeface, but that is handled by the document class file. Take care with the use of[3] the curly braces in typeface changes; they mark the beginning and end of the text that is to be in the different typeface.

[1]This is the second footnote. It starts a series of three footnotes that add nothing informational, but just give an idea of how footnotes work and look. It is a wordy one, just so you see how a longish one plays out.

[2]A third footnote, here. Let's make this a rather short one to see how it looks.

[3]A fourth, and last, footnote.

You can use whatever symbols, accented characters, or non-English characters you need anywhere in your document; you can find a complete list of what is available in the *LaTeX User's Guide*[?].

2.2 Math Equations

You may want to display math equations in three distinct styles: inline, numbered or non-numbered display. Each of the three are discussed in the next sections.

2.2.1 Inline (In-text) Equations

A formula that appears in the running text is called an inline or in-text formula. It is produced by the **math** environment, which can be invoked with the usual `\begin. . .\end` construction or with the short form `$. . .$`. You can use any of the symbols and structures, from α to ω, available in LaTeX[?]; this section will simply show a few examples of in-text equations in context. Notice how this equation: $\lim_{n\rightarrow \infty}x=0$, set here in in-line math style, looks slightly different when set in display style. (See next section).

2.2.2 Display Equations

A numbered display equation – one set off by vertical space from the text and centered horizontally – is produced by the **equation** environment. An unnumbered display equation is produced by the **displaymath** environment.

Again, in either environment, you can use any of the symbols and structures available in LaTeX; this section will just give a couple of examples of display equations in context. First, consider the equation, shown as an inline equation above:

$$\lim_{n\rightarrow \infty}x=0 \tag{1}$$

Notice how it is formatted somewhat differently in the **displaymath** environment. Now, we'll enter an unnumbered equation:

$$\sum_{i=0}^{\infty} x + 1$$

and follow it with another numbered equation:

$$\sum_{i=0}^{\infty}x_i=\int_{0}^{\pi+2} f \tag{2}$$

just to demonstrate LaTeX's able handling of numbering.

2.3 Citations

Citations to articles [?, ?, ?, ?], conference proceedings [?] or books [?, ?] listed in the Bibliography section of your article will occur throughout the text of your article. You should use BibTeX to automatically produce this bibliography; you simply need to insert one of several citation commands with a key of the item cited in the proper location in the `.tex` file [?]. The key is a short reference you invent to uniquely identify each work; in this sample document, the key is the first author's surname and a word from the title. This identifying key is included with each item in the `.bib` file for your article.

The details of the construction of the `.bib` file are beyond the scope of this sample document, but more information can be found in the *Author's Guide*, and exhaustive details in the *LaTeX User's Guide*[?].

Table 1: Frequency of Special Characters

Non-English or Math	Frequency	Comments
Ø	1 in 1,000	For Swedish names
π	1 in 5	Common in math
\$	4 in 5	Used in business
Ψ^2_1	1 in 40,000	Unexplained usage

Figure 1: A sample black and white graphic.

This article shows only the plainest form of the citation command, using `\cite`. This is what is stipulated in the SIGS style specifications. No other citation format is endorsed or supported.

2.4 Tables

Because tables cannot be split across pages, the best placement for them is typically the top of the page nearest their initial cite. To ensure this proper "floating" placement of tables, use the environment **table** to enclose the table's contents and the table caption. The contents of the table itself must go in the **tabular** environment, to be aligned properly in rows and columns, with the desired horizontal and vertical rules. Again, detailed instructions on **tabular** material is found in the *LaTeX User's Guide*.

Immediately following this sentence is the point at which Table 1 is included in the input file; compare the placement of the table here with the table in the printed dvi output of this document.

To set a wider table, which takes up the whole width of the page's live area, use the environment **table*** to enclose the table's contents and the table caption. As with a single-column table, this wide table will "float" to a location deemed more desirable. Immediately following this sentence is the point at which Table 2 is included in the input file; again, it is instructive to compare the placement of the table here with the table in the printed dvi output of this document.

2.5 Figures

Like tables, figures cannot be split across pages; the best placement for them is typically the top or the bottom of the page nearest their initial cite. To ensure this proper "floating" placement of figures, use the environment **figure** to enclose the figure and its caption.

This sample document contains examples of **.eps** files to be displayable with LaTeX. If you work with pdfLaTeX, use files in the **.pdf** format. Note that most modern TeX system will convert **.eps** to **.pdf** for you on the fly. More details on each of these is found in the *Author's Guide*.

As was the case with tables, you may want a figure that spans two columns. To do this, and still to ensure proper "floating" placement of tables, use the environment **figure*** to enclose the figure and its caption. and don't forget to end the environment with figure*, not figure!

2.6 Theorem-like Constructs

Figure 2: A sample black and white graphic that has been resized with the `includegraphics` command.

Table 2: Some Typical Commands

Command	A Number	Comments
`\alignauthor`	100	Author alignment
`\numberofauthors`	200	Author enumeration
`\table`	300	For tables
`\table*`	400	For wider tables

Figure 3: A sample black and white graphic that needs to span two columns of text.

Figure 4: A sample black and white graphic that has been resized with the `includegraphics` command.

Other common constructs that may occur in your article are the forms for logical constructs like theorems, axioms, corollaries and proofs. There are two forms, one produced by the command `\newtheorem` and the other by the command `\newdef`; perhaps the clearest and easiest way to distinguish them is to compare the two in the output of this sample document:

This uses the **theorem** environment, created by the `\newtheorem` command:

THEOREM 1. *Let f be continuous on $[a, b]$. If G is an antiderivative for f on $[a, b]$, then*

$$\int_a^b f(t)dt = G(b) - G(a).$$

The other uses the **definition** environment, created by the `\newdef` command:

Definition 1. If z is irrational, then by e^z we mean the unique number which has logarithm z:

$$\log e^z = z$$

Two lists of constructs that use one of these forms is given in the *Author's Guidelines*.

There is one other similar construct environment, which is already set up for you; i.e. you must *not* use a `\newdef` command to create it: the **proof** environment. Here is a example of its use:

PROOF. Suppose on the contrary there exists a real number L such that

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = L.$$

Then

$$l = \lim_{x\to c} f(x) = \lim_{x\to c} \left[gx \cdot \frac{f(x)}{g(x)} \right] = \lim_{x\to c} g(x) \cdot \lim_{x\to c} \frac{f(x)}{g(x)} = 0 \cdot L = 0,$$

which contradicts our assumption that $l \neq 0$. $\square$

Complete rules about using these environments and using the two different creation commands are in the *Author's Guide*; please consult it for more detailed instructions. If you need to use another construct, not listed therein, which you want to have the same formatting as the Theorem or the Definition[**?**] shown above, use the `\newtheorem` or the `\newdef` command, respectively, to create it.

A *Caveat* for the TEX Expert

Because you have just been given permission to use the `\newdef` command to create a new form, you might think you can use TEX's `\def` to create a new command: *Please refrain from doing this!* Remember that your LATEX source code is primarily intended to create camera-ready copy, but may be converted to other forms – e.g. HTML. If you inadvertently omit some or all of the `\def`s recompilation will be, to say the least, problematic.

3. CONCLUSIONS

This paragraph will end the body of this sample document. Remember that you might still have Acknowledgments or Appendices; brief samples of these follow. There is still the Bibliography to deal with; and we will make a disclaimer about that here: with the exception of the reference to the LATEX book, the citations in this paper are to articles which have nothing to do with the present subject and are used as examples only.

4. ACKNOWLEDGMENTS

This section is optional; it is a location for you to acknowledge grants, funding, editing assistance and what have you. In the present case, for example, the authors would like to thank Gerald Murray of ACM for his help in codifying this *Author's Guide* and the **.cls** and **.tex** files that it describes.

APPENDIX

A. HEADINGS IN APPENDICES

The rules about hierarchical headings discussed above for the body of the article are different in the appendices. In the **appendix** environment, the command **section** is used to indicate the start of each Appendix, with alphabetic order designation (i.e. the first is A, the second B, etc.) and a title (if you include one). So, if you need hierarchical structure *within* an Appendix, start with **subsection** as the highest level. Here is an outline of the body of this document in Appendix-appropriate form:

A.1 Introduction

A.2 The Body of the Paper

A.2.1 Type Changes and Special Characters

A.2.2 Math Equations

Inline (In-text) Equations.

Display Equations.

A.2.3 Citations

A.2.4 Tables

A.2.5 Figures

A.2.6 Theorem-like Constructs

A Caveat for the TEX Expert

A.3 Conclusions

A.4 Acknowledgments

A.5 Additional Authors

This section is inserted by LATEX; you do not insert it. You just add the names and information in the `\addition-alauthors` command at the start of the document.

A.6 References

Generated by bibtex from your .bib file. Run latex, then bibtex, then latex twice (to resolve references) to create the .bbl file. Insert that .bbl file into the .tex source file and comment out the command `\thebibliography`.

B. MORE HELP FOR THE HARDY

The sig-alternate.cls file itself is chock-full of succinct and helpful comments. If you consider yourself a moderately experienced to expert user of LATEX, you may find reading it useful but please remember not to change it.