Verification of Autonomous Systems

Alessandro Pinto
United Technologies Research Center, Berkeley, CA

This document does not contain any export controlled technical data.
Outline

• Introduction
• Verification and autonomous systems
• Thesis: Verify what the system knows and understands rather than what the system does
• Our contribution in the DARPA ALIAS program
United Technologies Business Units

Otis

UTC Climate, Controls & Security

Pratt & Whitney

UTC Aerospace Systems

2014 Sales: $65.1 billion
Segments: 45% Commercial & Industrial, 55% Aerospace

- Sikorsky: 11%
- UTC Climate, Controls, & Security: 25%
- UTC Aerospace Systems: 22%
- Otis: 20%
- Pratt & Whitney: 22%

This page does not contain any export controlled technical data.
Applications

UTRC interest

Autonomous rotorcraft

SARA

Pilot

Field Operator

GCS

ISR missions

Source: wikipedia

Source: cloudcaptech.com

Source: cloudcaptech.com

Source: cloudcaptech.com

UTAS EFB

Source: utasaerospace.com

Source: utasaerospace.com

Source: ccs.utc.com

Intelligent Buildings

Source: ccs.utc.com

Comfort

Efficiency

Safety
Verification

*General definition*

Verification methodology and algorithms

Model → Verification methodology and algorithms → Property

Y/N (Counterexample)
Verification

The known case

Verification methodology and algorithm
- Model checking
- Testing
- ...

Model
- Simulink
- Code
- ...

Property
- Requirements
- Common sense
- ...

Y/N (Counterexample)
Verification

Autonomous systems

Model? (Environment?, Perceptions?, Decision Making?)

Verification methodology and algorithm? (can't be define until we know Model and Property)

Property? (Not designed to do one thing)

Y/N (Counterexample)

A. Pinto, UTRC, pintoa@utrc.utc.com

This page does not contain any export controlled technical data.
Traditional System Design

From CONOPS to Operations

Running system

Scenarios describing environment and use of the system

Concept of Operations

Requirements and Architecture

Detailed Design

System Verification and Validation

Integration, Test, Verification

Experts + Systems engineers
Autonomous Systems

*Operations on CONOPS*

Mission objectives, constraints, relevant facts (function of time)

Needs to know/understand the subject matter
Knowledge-Based Systems

Archetype

Mission objectives, constraints, relevant facts (function of time)

From the environment

Perception → Knowledge-Base → Executive

→ Planning
→ Monitoring
→ Validation
→ Learning

To the environment
Knowledge Capturing and Verification

Presented by Alessandro Pinto
United Technologies Research Center, Berkeley, CA

This research was funded by the Defense Advanced Research Projects Agency (DARPA) Aircrew Labor In-Cockpit Automation System (ALIAS) program. The views, opinions and/or findings expressed are those of the author(s) and should not be interpreted as the official views or policies of the Department of Defense or the U.S. Government.

Distribution Statement "A" (Approved for Public Release, Distribution Unlimited)
Completeness

**General definition**

Is there a query $\Phi$ such that:

- $\text{Knowledge Base (KB)} \vdash \Phi$?
- $\text{Knowledge Base (KB)} \vdash \neg \Phi$?

<table>
<thead>
<tr>
<th>Incomplete</th>
<th>yes</th>
<th>complete</th>
</tr>
</thead>
</table>

$\text{KB} \not\models \Phi$?
Completeness

Examples of knowledge bases

$$A \Rightarrow B$$

Yes/No

Examples of knowledge bases

$$K B$$

Formulae over $v = value$

Syntax (prop. logic)

Semantic (validity)

KB

Knowledge graph

Execution Engine

Static semantics
Completeness of Knowledge Graphs

Syntax, semantics, and structural rules

- Inputs are available
- Some variables are constant and fixed
- Everything else is "unknown" or non-det.

Assume query $\Phi \equiv v = value$?

If $v$ is non-det., then KB is incomplete

Rules:
- No cycles
- No leaf nodes
- Nodes with common child update different variables
- All variables are updated by some node
- All variables are read
- ...

N1

att1: T1
att2: T2

(Java object)

N2

att1: T1
att3: T3

If (cond) then update

(Java statement)

Engine runs

$V \rightarrow V^+$

Env. Inputs

Analyze dependencies → Build abstract graph → Check rules → Generate report
Completeness of Knowledge Graphs

Integration in the knowledge modeling tool
Lessons learned

• Knowledge graphs easy to understand by software engineers
• Not as powerful as declarative knowledge
• Verification is hard on generic code leading to spurious error reports (conservative abstraction)
• If fast, verification can be integrated in the knowledge modeling process
• Verification needs to be precise: avoid confusing users and loosing trust
Process

From knowledge sources to execution

Raw sources → Automatic knowledge extraction

Model idea

Formal knowledge model

Formal verification → No problems

Formal compilation

To knowledge engine

Knowledge tests

Problems
Tools

An integrated environment for knowledge modeling and verification

Mix of powerful textual and graphical editors

Model browser

Rapid access to verification and code generation tools

Verification task manager for model debugging
Conclusions

- Autonomous systems solve new class of problems
- A new design and verification method is needed for autonomous systems
- Verification should focus on what these systems know and understand