MITL Specification Debugging for Monitoring of Cyber-Physical Systems

Adel Dokhanchi, Bardh Hoxha and **Georgios Fainekos**

1st Workshop on Monitoring and Testing of Cyber-Physical Systems

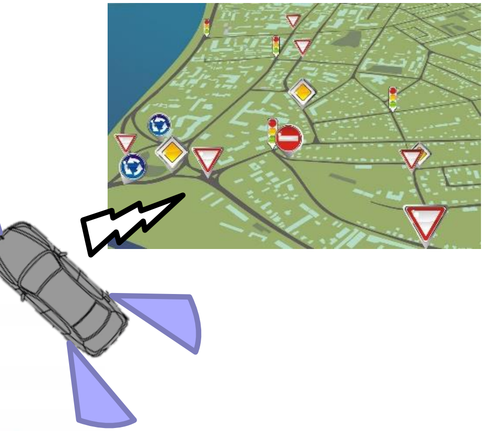
April 2016

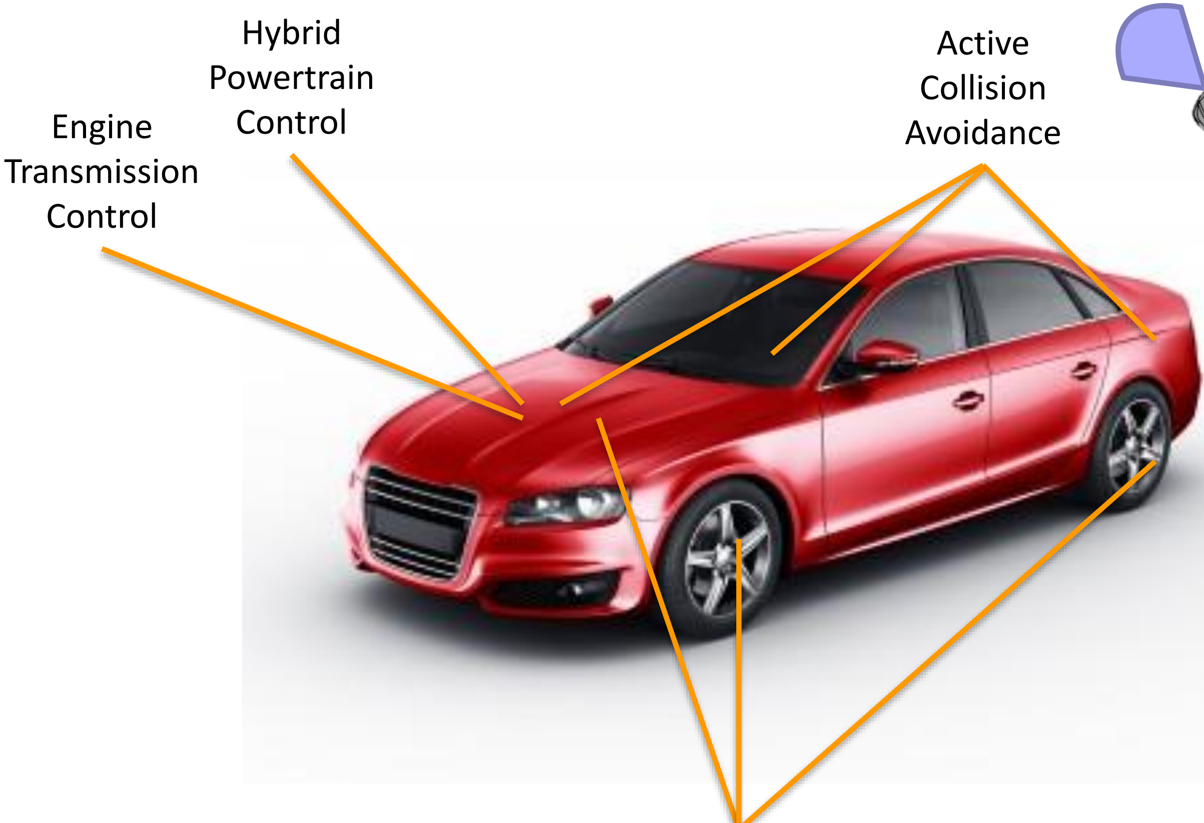
School of Computing, Informatics and
Decision System Engineering

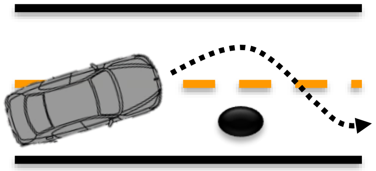
Arizona State University

✉ fainekos at asu.edu

Modern Vehicles




Hybrid Powertrain Control
Engine Transmission Control
Active Collision Avoidance



Electronic Stability Control

<u>Already demonstrated:</u>

- Lane following
- Fully autonomous driving

Trust? : Sampling of automotive recalls (~2011-12) due to software errors …

- "A software err … fting from 5th to 4th gear when c … his may result in decreased engi … "
- … the software that "allows the ECU to establish a 'handshake' with the engine is in error. The ECU monitors ce … ut of tolerance, the software pi … code. As the ECU tries to find an … ugh idle or stalling situation ensues."
- … to up … stances, it is possib … on opposite
- If the fault occu … while driving - which would m … steering. Braking or pressing the
- …

No downshifting from 5th to 4th

Rough idling or stalling due to complicated adaptive ECU

Electric motor to rotate in the direction opposite to that selected by the transmission

Cruise control does not disengage unless turning off the ignition

Many more …

Trust? : Sampling of automotive recalls (~2011-12) due to software errors …

- "A software e fting from 5th to 4th gear whe his may result in decreased en "

Under C(x,p) conditions the system should always switch from 5th to 4th gear.

- … the software that “allows the ECU to establish a ‘handshake’ with the engine is in error. The ECU monitors out of tolerance, the software p t code. As the ECU tries to find an ough idle or stalling situation ensues.”

The engine should never stall while idle.

- … to up stances, it is possib n opposite

The electric motor should always rotate in the direction selected by the transmission.

- If the fault hile driving - which woul eering. Braking or pressing

The cruise control should always disengage when the “turn off” button is pressed.

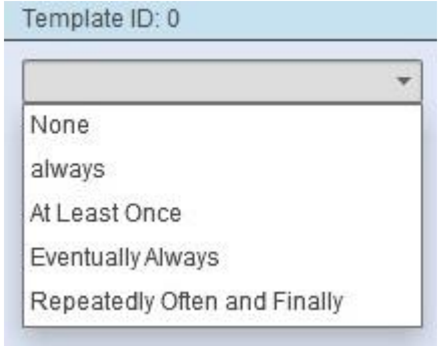
- …

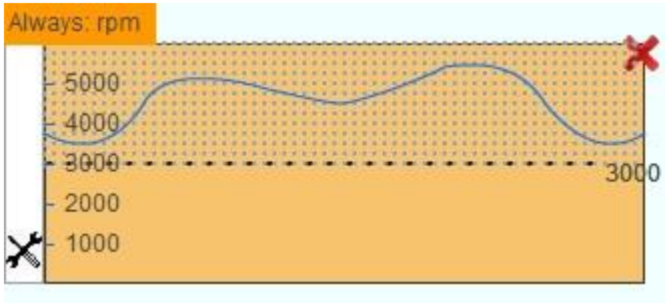
Formal Specification

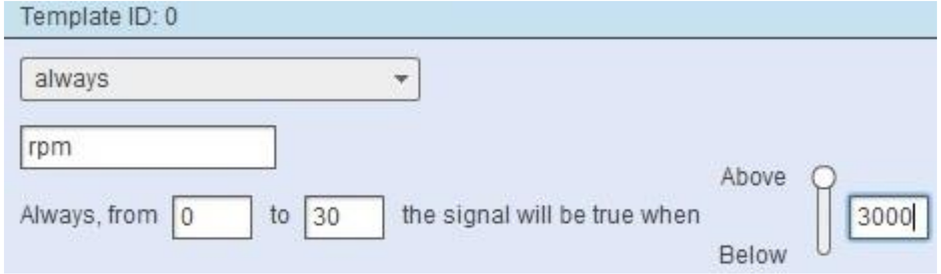
- Natural Language is not appropriate for verification
  - Ambiguous, inaccurate, inconsistent?
- Alternative: use mathematical logic
  - Elicitation process: Challenging & Error prone
- Validate the specification
  - Verification with wrong specification is useless
- Before verification we need to analyze the specification

Previous Work: ViSpec

- ViSpec helps transforming Pre-specified templates in NL
- Easy to use
- No need for MITL background






Template ID: 0

always

rpm

Always, from 0 to 30 the signal will be true when Above / Below 3000

- B. Hoxha and H. Bach and H. Abbas and A. Dokhanchi and Y. Kobayashi and G. Fainekos, **Towards Formal Specification Visualization for Testing and Monitoring of Cyber-Physical Systems,** DIFTS 2014
- B. Hoxha and N. Mavridis and Georgios Fainekos, **VISPEC: A graphical tool for easy elicitation of MTL requirements,** IROS 2015

ViSpec – Specification Classes

Safety:

$$\Box_I \phi$$

Reachability:

$$\Diamond_I \phi$$

Stabilization:

$$\Diamond_I \Box_I \phi$$

Recurrence:

$$\Box_I \Diamond_I \phi$$

Implication:

$$\phi \rightarrow \psi$$

Reactive Response:

$$\Box_I (\phi \rightarrow M_I \psi)$$

Conjunction:

$$\phi \wedge \psi$$

Non-strict Sequencing:

$$N_I (\phi \wedge M_I \psi)$$

$$M \in \{\Box, \Diamond\}, N \in \{\Box, \Diamond\}$$

Motivating Example: On-Line Survey

We asked:

"At some time in the first 30 seconds, the vehicle speed (v) will go over 100 and stay above 100 for 20 seconds"

Response:

$$\varphi = \Diamond_{[0,30]}(\,(v > 100) \Rightarrow \Box_{[0,20]}(v > 100)\,)$$

φ is a tautology

- $(v > 100) = \bot$ any time in [0,30]

$$((v > 100) \Rightarrow \Box_{[0,20]}(v > 100)) = \top$$

- $(v > 100) = \top$ all the time in [0,30]

$\Box_{[0,20]}(v > 100) = \top$between [0,10]

$((v > 100) \Rightarrow \Box_{[0,20]}(v > 100)) = \top$ between [0,10]

- B. Hoxha and N. Mavridis and Georgios Fainekos, VISPEC: A graphical tool for easy elicitation of MTL requirements, IROS 2015

Problem Formulation

Problem 1 (System Independent MITL Analysis):

Given an MITL formula φ, find whether φ has any of the following logical issues:

- Validity: the specification is unsatisfiable or a tautology.
- Redundancy: the formula has redundant conjuncts.
- Vacuity: some subformulas do not contribute to the satisfiability of the formula.

Problem 2 (System Dependent Vacuity Checking):

Given an MITL formula φ, and signal μ, check whether μ satisfies the antecedent failure mutation of φ.

1. **A. Dokhanchi, B. Hoxha, and G. Fainekos, *Metric interval temporal logic specification elicitation and debugging*. MEMOCODE 2015, Austin, TX, USA**
2. **Extension of 1 under review**

Contributions

- We present a specification debugging algorithm for a fragment of Metric Interval Temporal Logic (MITL) specifications and, consequently for Signal Temporal Logic (STL).
  - We extend Linear Temporal Logic (LTL) vacuity detection algorithms to real-time specifications in MITL.
- We provide a signal vacuity detection algorithm to indicate to the testing team the signals that vacuously satisfy the specification.
- We present experimental results on specifications that typically appear in CPS specifications.

Overview

- Motivation
- Preliminaries
- System Independent MITL Analysis
- System Dependent Vacuity Checking
- Experiments
- Conclusion & Future Research

Metric Interval Temporal Logic: Semantic Intuition

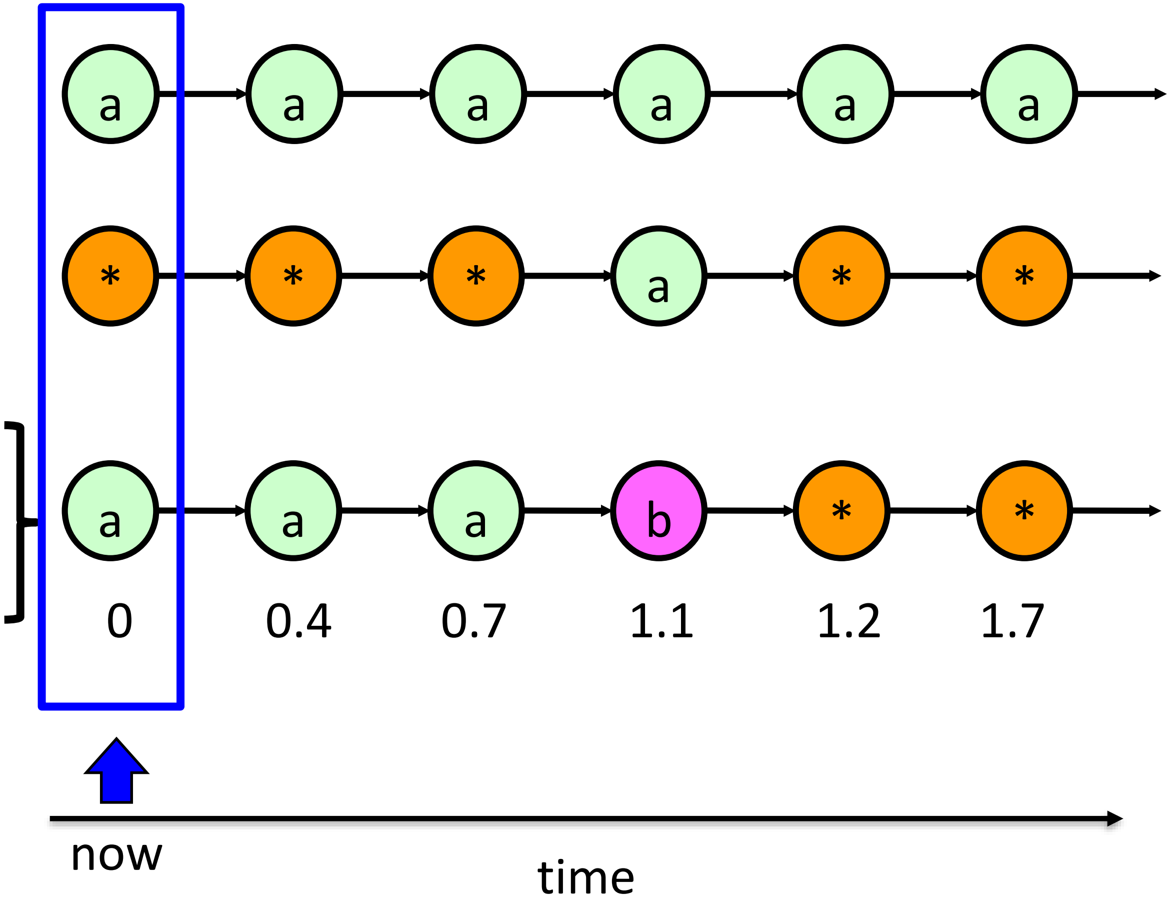
$\phi \ ::= \top \ | \, p \, | \ \neg\phi \, | \, \phi_1 \vee \phi_2 \, | \, \Box_I \phi \, | \, \Diamond_I \phi \, | \, \phi_1 U_I \phi_2$

$\Box a$- always a

$\Diamond_{[1,3]}$a-eventually a

$a \; U \; b$- a until b

$a \; U_{[1,1.5]} \; b$ -a until b



Subset of MITL

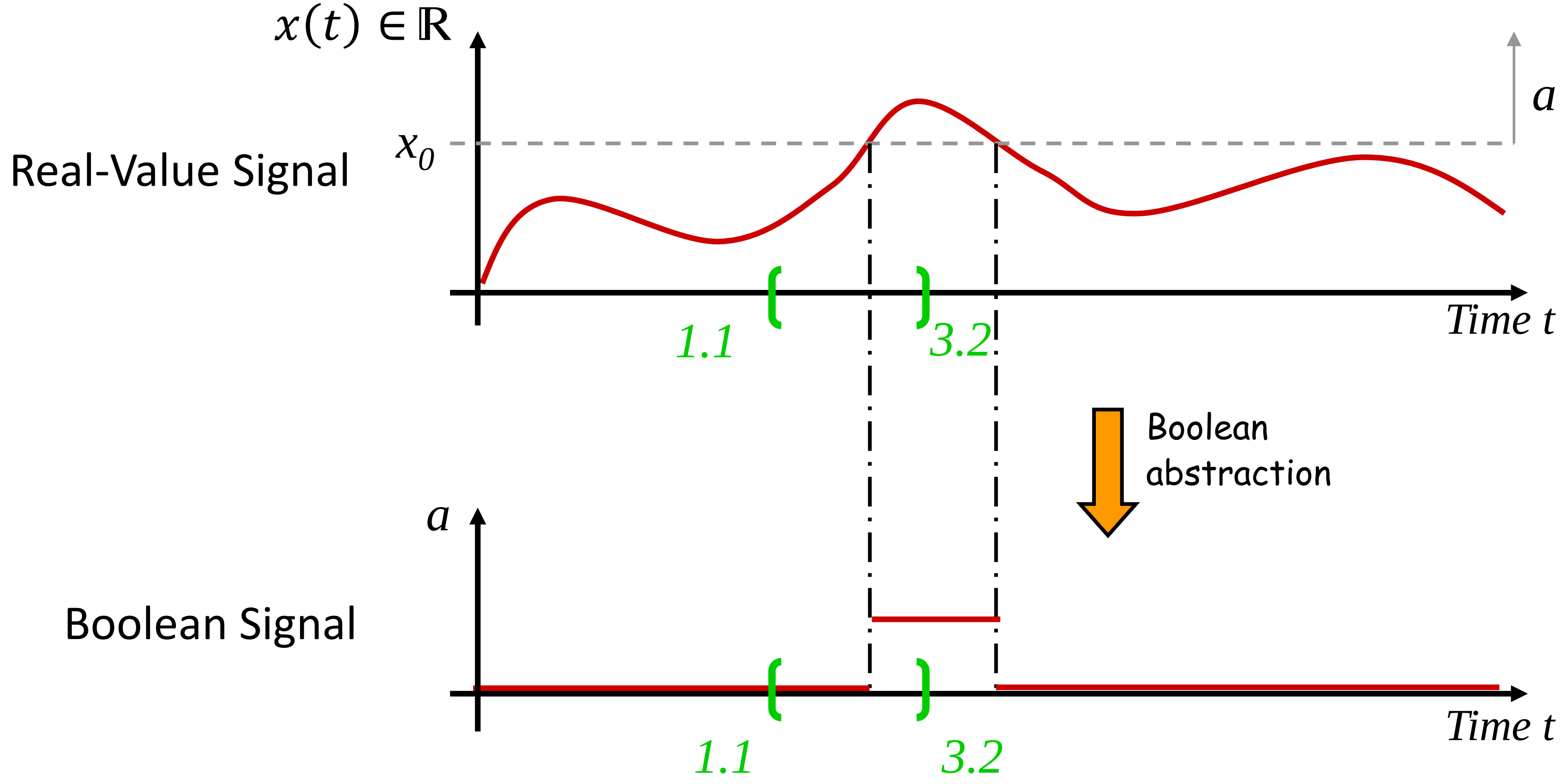
- Bounded-MITL(◇, □) with only Always & Eventually operator
- Negation Normal Form
- Syntax:

$$\phi \ ::= \top \mid \bot \mid p \mid \neg p \mid \phi_1 \vee \phi_2 \mid \phi_1 \wedge \phi_2 \mid \Box_I \phi \ \mid \Diamond_I \phi$$

- No Until, Release, Next operator is used

Signal Temporal Logic

Specification example: $\Box_{[1.1,3.2]}(x(t) \geq x_0)$

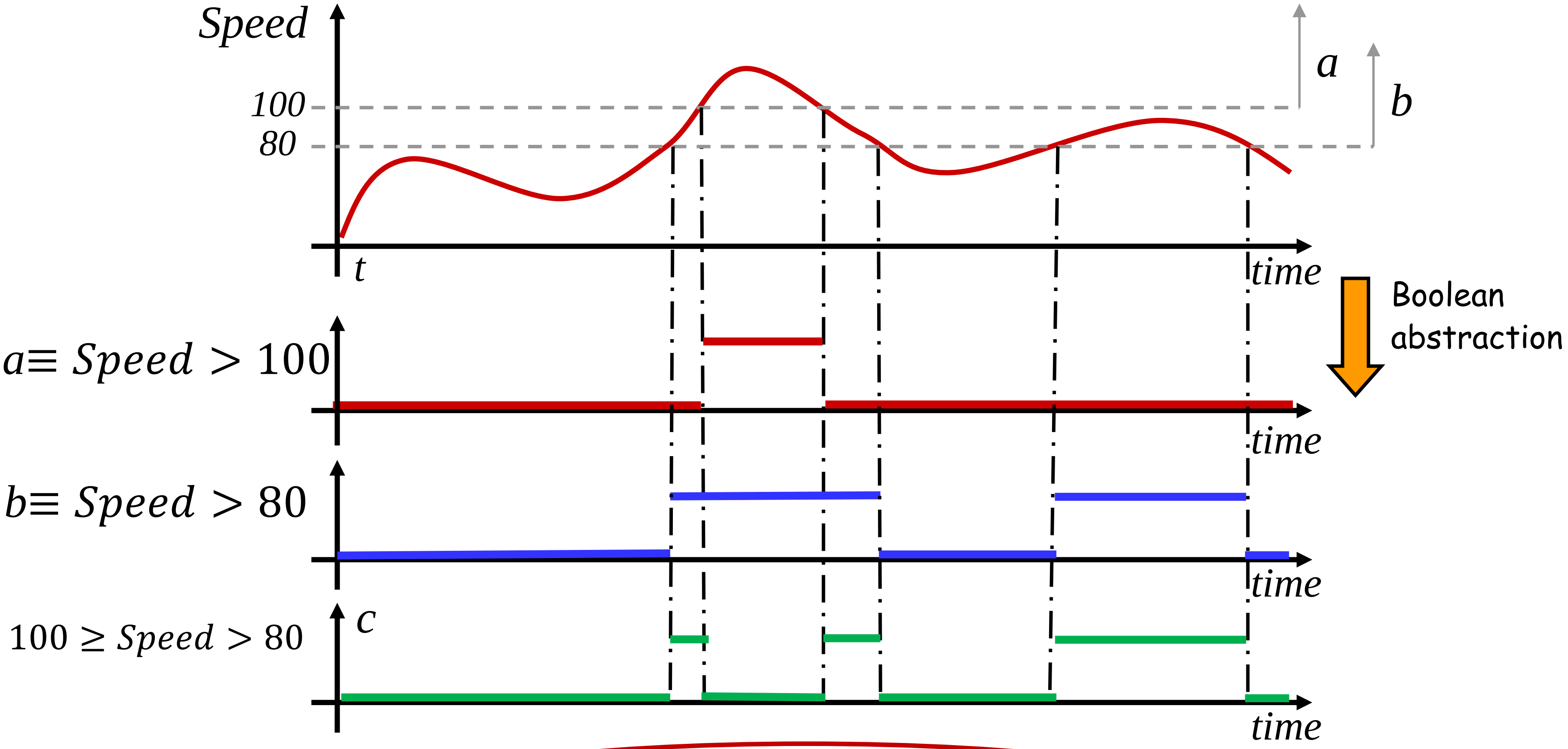


Notice example is MITL if we replace the predicate with a proposition: $a \equiv (x(t) \geq x_0)$

Overview

- Motivation
- Preliminaries
- System Independent MITL Analysis
- System Dependent Vacuity Checking
- Experiments
- Conclusion & Future Research

Transforming STL to MITL



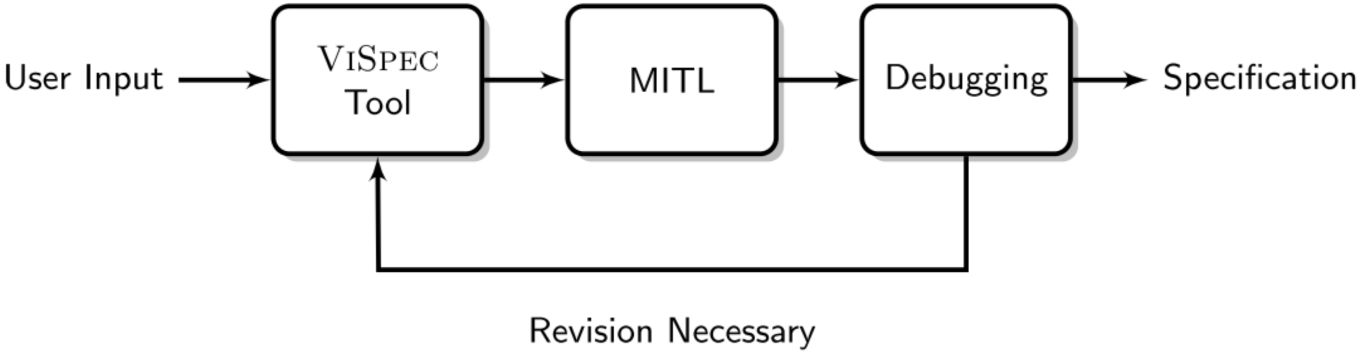
$$(speed > 100) \Rightarrow (speed > 80)$$

$$a \overset{?}{\Rightarrow} b$$

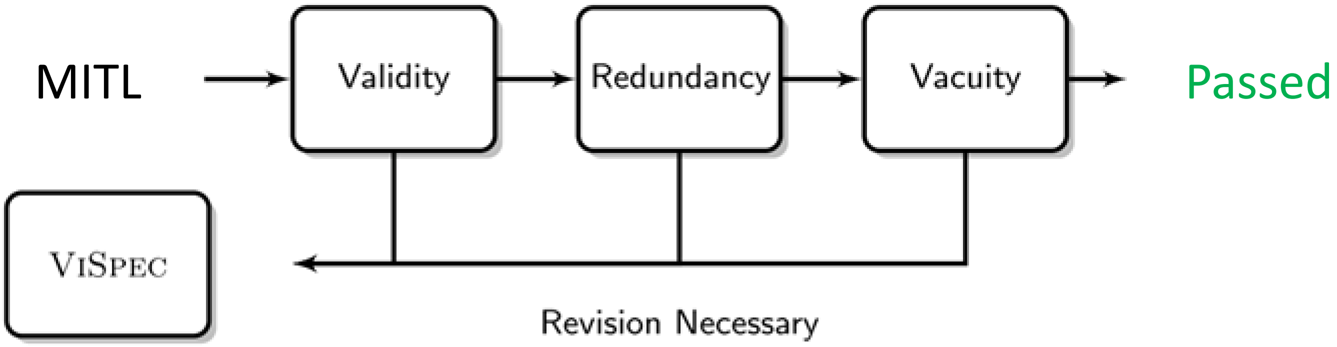
$$a \Rightarrow a \vee c$$

Debugging MITL Specification

Specification Elicitation Framework



3-Levels of Specification Debugging



Validity Issues Detection

Checking whether φ is unsatisfiable or a tautology

A valid formula is one where φ and $\neg\varphi$ are satisfiable

We asked:

"At some time in the first 30 seconds, the vehicle speed (v) will go over 100 and stay above 100 for 20 seconds"

Response:

$$\varphi = \Diamond_{[0,30]}(\ (v > 100) \Rightarrow \Box_{[0,20]}(v > 100)\ )$$

φ is a tautology

Redundancy Issues Detection

Conjunctive formula: $\Phi = \bigwedge_{j=1}^{k} \varphi_j$

Removing conjunct:

$$\bigwedge_{j=1}^{i-1} \varphi_j \ \wedge \bigwedge_{j=i+1}^{k} \varphi_j \equiv \Phi \backslash \varphi_i$$

If $\exists\ \varphi_i$

$\Phi \backslash \varphi_i \vDash \varphi_i$

Then φ_i is redundant

Example $\varphi_2 = p \wedge \square_{[0,10]} p$ → $\square_{[0,10]} p \vDash p$

Algorithm 1: Checks $\Phi \backslash \varphi_i \vDash \varphi_i$ for each conjunct

Creates a list of redundant conjuncts

- **H. Chockler and O. Strichman, Before and after vacuity. Form. Methods Syst. Des., 34(1):37–58, Feb. 2009.**

Redundancy Example

CPS example

$$\varphi = \Diamond_{[0,30]}\xcancel{(speed > 100)} \wedge \Diamond_{[0,20]}(speed > 100)$$

$\Diamond_{[0,30]}(v > 100)$ is redundant since

$$\Diamond_{[0,20]}(v > 100) \vDash \Diamond_{[0,30]}(v > 100)$$

Vacuity Issues Detection

If sub-formula $\psi \in \varphi$ does not affect the satisfiability of φ, then φ is vacuous

Remove ψ

Vacuous specifications are **equivalent** to their mutant

H. Chockler and O. Strichman, Before and after vacuity. Form. Methods Syst. Des., 34(1):37–58, Feb. 2009.

Mutation of MITL for Vacuity Checking

Mutation with assigning ⊥ to literal occurrence

$$\varphi = (\neg p \wedge q) \vee \Diamond_{[0,10]} p \vee \Box_{[0,10]} q$$

$$\varphi[\neg p \leftarrow \bot] = (\bot \wedge q) \vee \Diamond_{[0,10]} p \vee \Box_{[0,10]} q$$

4 literal occurrence => 4 mutation

Algorithm 2: Checks $\Phi \vDash \varphi_i[l \leftarrow \bot]$ for each mutation

Creates a list of mutated sub-formulas

Vacuity Theorem

Theorem (MITL Vacuity with respect to Specification): Assume that the specification Φ is a conjunction of MITL formulas. If $\exists \varphi_i \in \Phi$ and $\exists\ l \in$ litOccur(φ_i), such that $\Phi \vDash \varphi_i\ [l \leftarrow \bot]$, then Φ satisfies φ_i vacuously ($\Phi \vDash_v \varphi_i$).

We proved that in Φ

if $\exists$ a conjunct φ_i & literal occurrence $l \in$ litOccur(φ_i)

s.t $\Phi \vDash \varphi_i[l \leftarrow \bot]$

then Φ is inherently vacuous

A. Dokhanchi, B. Hoxha, and G. Fainekos, *Metric interval temporal logic specification elicitation and debugging*. MEMOCODE 2015, Austin, TX, USA,

Vacuity Example

CPS example

$$\varphi_{STL} = \Diamond_{[0,10]}((speed > 100) \wedge \Diamond_{[0,10]}(speed > 80))$$

$\varphi = \Diamond_{[0,10]}(a \wedge \Diamond_{[0,10]}\, b)$ is not vacuous

However…

$\varphi' = \Diamond_{[0,10]}(a \wedge \Diamond_{[0,10]}(a \vee c))$ is vacuous

Where $a: (speed > 100)$ and $c: (100 \geq speed > 80)$

$$\varphi' \vDash \Diamond_{[0,10]}(a \wedge \Diamond_{[0,10]}(a \vee \bot))$$

Overview

- Motivation
- Preliminaries
- System Independent MITL Analysis
- System Dependent Vacuity Checking
- Experiments
- Conclusion & Future Research

Vacuous Signals

- The MITL specification

$$\varphi = \square_{[0,5]}(\,(request) \Rightarrow \Diamond_{[0,10]}(acknowledge)\,)$$

- φ is passed the MITL Specification Debugging Framework

- Any signal μ that does not satisfy $request$ at any point in time will vacuously satisfy φ.

- Signals that do not satisfy the ***antecedent*** (***precondition***) of the subformula are called vacuous signals.

- Vacuous Signals satisfy antecedent failure mutation of φ

Antecedent Failure Mutation

- For each implication (φ ⇒ ψ),
  (φ) is the precondition (antecedent) of the implication.

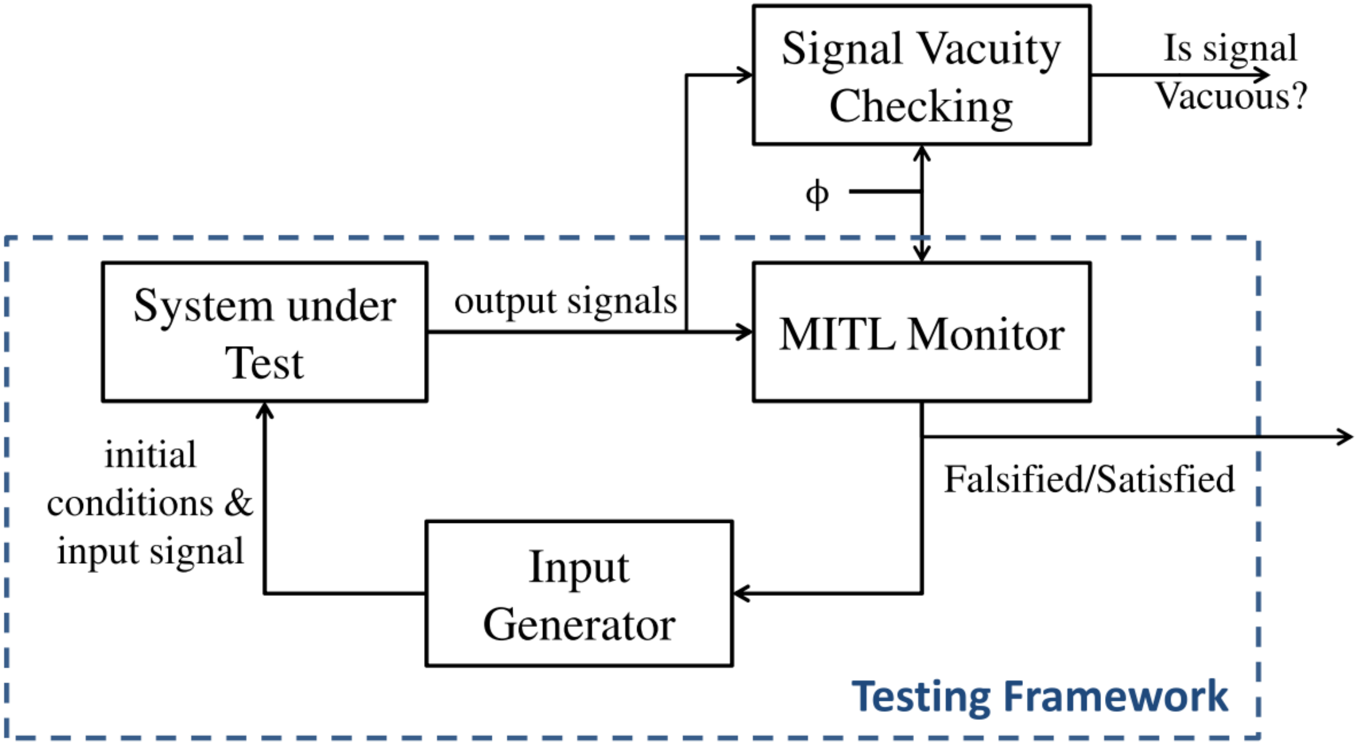
- Antecedent Failure Mutation is the assertion that the precondition (φ) never happens.

- Example:

  Antecedent Failure of φ is ¬φ

- Signals that satisfy ¬φ are vacuous signals

Vacuity Detection in Testing



Overview

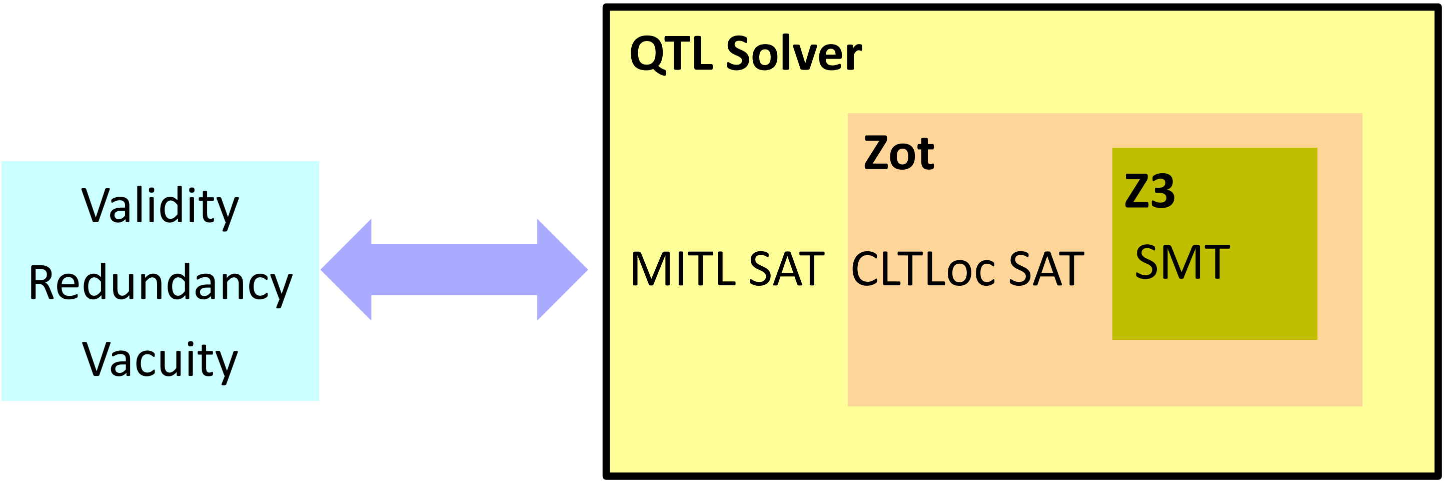
- Motivation
- Preliminaries
- System Independent MITL Analysis
- System Dependent Vacuity Checking
- Experiments
- Conclusion & Future Research

Implementation and Experiments

We used MITL satisfiability solver for this method:

$\varphi \vDash \psi$ iff $(\varphi \Longrightarrow \psi) \equiv T$ iff $(\neg\varphi \vee \psi) \equiv T$ iff $(\varphi \wedge \neg\psi) \equiv \bot$

We detected all the issues with MITL SAT solver

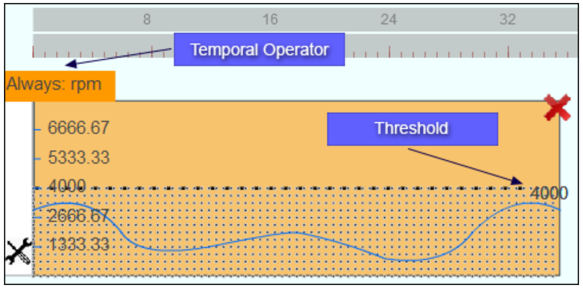


M. Bersani and M. Rossi and P. San Pietro, A tool for deciding the satisfiability of continuous-time metric temporal logic. Acta Informatica, pages 1–36, 2015.

ViSpec – Usability Study

Each user received ten tasks:

- To formalize a NL specification in automotive industry through ViSpec



Group I: Non-expert users

No experience in working with requirements.

20 subjects from the student community at ASU

Group 2: Expert users

Experienced in working with requirements (not necessarily formal requirements)

10 subjects from the industry in the Phoenix area

B. Hoxha and N. Mavridis and Georgios Fainekos, VISPEC: A graphical tool for easy elicitation of MTL requirements, IROS 2015

Specification Checks

Seven of ten tasks have no detected issue

Example task with erroneous specifications:

Stabilization “*At some point in time in the first 30 seconds, vehicle speed will go over 100 and stay above for 20 seconds*.”

Correct answer: $\Diamond_{[0,30]} \Box_{[0,20]}(p_1)$

p_1 : speed>100

Incorrect Answers:

Specification	Detected Error
$\Diamond_{[0,30]}(p_1) \wedge \Diamond_{[0,20]} (p_1)$	$\Diamond_{[0,30]}(p_1)$ is redundant
$\Diamond_{[0,30]}(p_1 \Rightarrow \Box_{[0,20]} (p_1))$	Tautology

Error in Oscillation Task

"At every point in time in the first 40 seconds, vehicle speed will go over 100 in the next 10 seconds."

Correct answer:

$$\Box_{[0,40]} \Diamond_{[0,10]}(p_1)$$

p_1 : speed>100

Incorrect answer (with Redundancy Error):

$$\Box_{[0,40]} (p_1) \wedge \Box_{[0,40]} \Diamond_{[0,20]}(p_1)$$

Issue:

$\Box_{[0,40]} \Diamond_{[0,20]}(p_1)$ is redundant

Error in Long Sequence Task

"If, at some point in time in the first 40 seconds, vehicle speed goes over 80 then from that point on, if within the next 20 seconds the engine speed goes over 4000, then, for the next 30 seconds, the vehicle speed should be over 100."

$$\Diamond_{[0,40]} (\ (\text{speed}>80) \Rightarrow \Diamond_{[0,20]}(\text{rpm}>4000 \Rightarrow \Box_{[0,30]}\text{speed}>100)\ )$$

↓(STL2MITL)

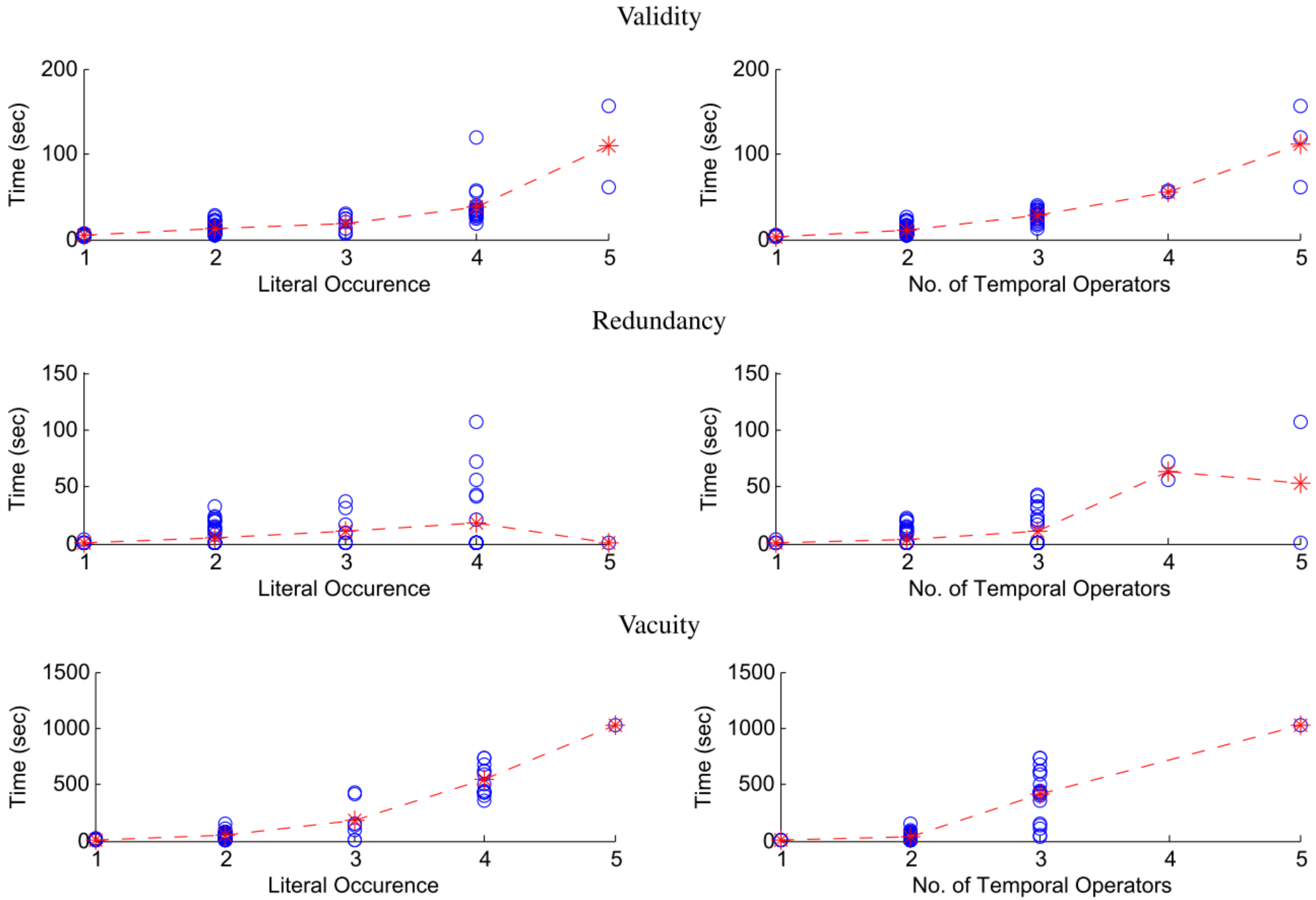
$$\Diamond_{[0,40]\ ((} p_1 \vee p_3) \Rightarrow \Diamond_{[0,20]} (p_2 \Rightarrow \Box_{[0,30]} p_1)\ )$$

p_1: speed>100 p_2 : rpm>4000 p_3: 100≥speed>80

Incorrect Answers:

Specification	Detected Error
$\Diamond_{[0,40]((} p_1 \vee p_3) \Rightarrow \Diamond_{[0,20]} p_2 \wedge \Box_{[0,30]} p_1)\ )$	Vacuous formula
$\Diamond_{[0,40](} p_1 \vee p_3) \wedge \Diamond_{[0,40]} p_2 \wedge \Diamond_{[0,40]} \Box_{[0,30]} p_1$	$\Diamond_{[0,40](} p_1 \vee p_3)$ is redundant

Runtime Overhead

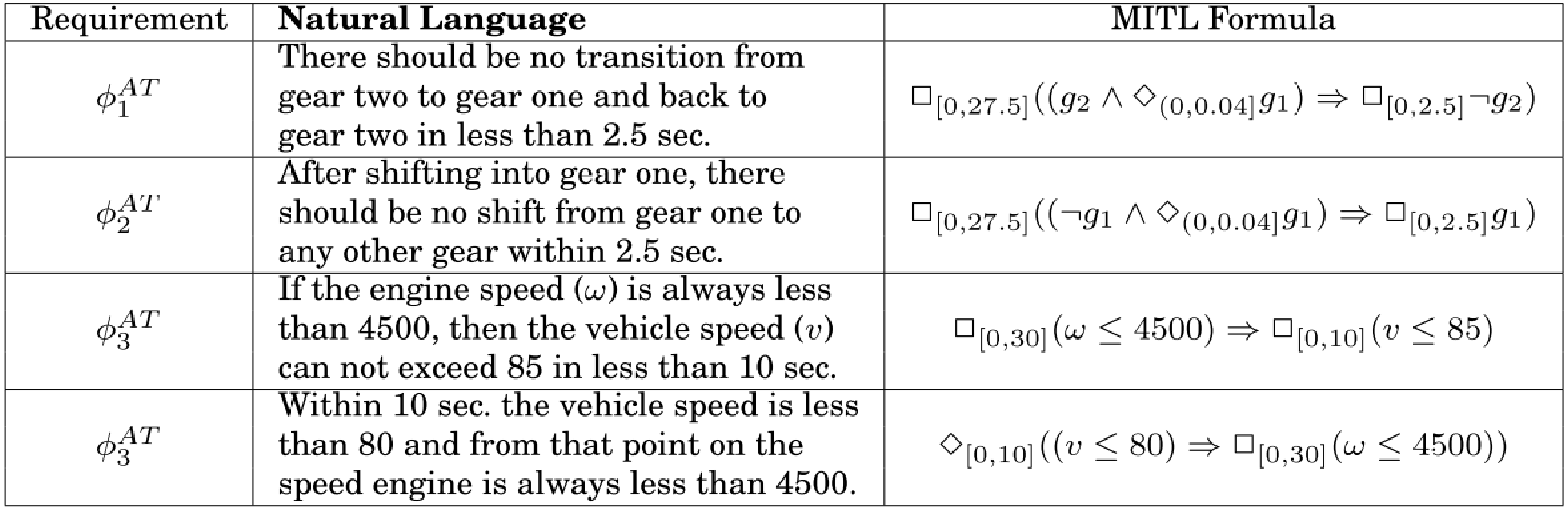


Validity

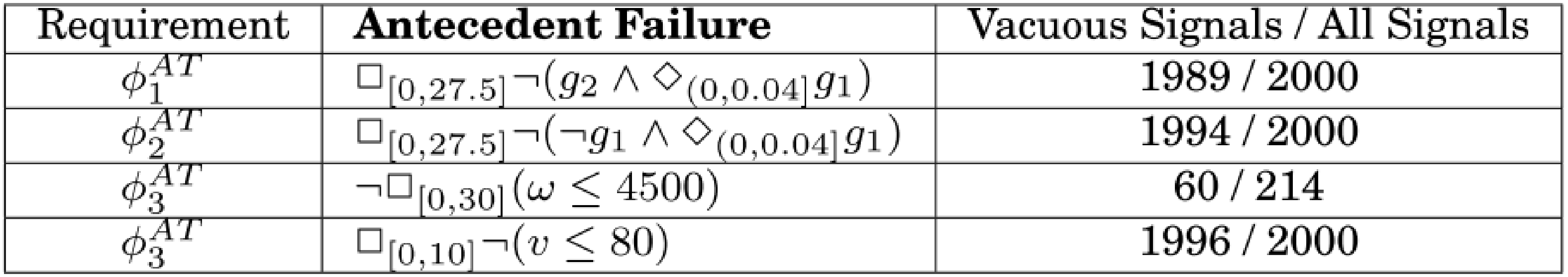
Redundancy

Vacuity

Antecedent Failure Detection



Requirement	**Natural Language**	MITL Formula
ϕ_1^{AT}	There should be no transition from gear two to gear one and back to gear two in less than 2.5 sec.	$\Box_{[0,27.5]}((g_2 \wedge \Diamond_{(0,0.04]} g_1) \Rightarrow \Box_{[0,2.5]} \neg g_2)$
ϕ_2^{AT}	After shifting into gear one, there should be no shift from gear one to any other gear within 2.5 sec.	$\Box_{[0,27.5]}((\neg g_1 \wedge \Diamond_{(0,0.04]} g_1) \Rightarrow \Box_{[0,2.5]} g_1)$
ϕ_3^{AT}	If the engine speed (ω) is always less than 4500, then the vehicle speed (v) can not exceed 85 in less than 10 sec.	$\Box_{[0,30]}(\omega \leq 4500) \Rightarrow \Box_{[0,10]}(v \leq 85)$
ϕ_3^{AT}	Within 10 sec. the vehicle speed is less than 80 and from that point on the speed engine is always less than 4500.	$\Diamond_{[0,10]}((v \leq 80) \Rightarrow \Box_{[0,30]}(\omega \leq 4500))$



Requirement	**Antecedent Failure**	Vacuous Signals / All Signals
ϕ_1^{AT}	$\Box_{[0,27.5]} \neg (g_2 \wedge \Diamond_{(0,0.04]} g_1)$	1989 / 2000
ϕ_2^{AT}	$\Box_{[0,27.5]} \neg (\neg g_1 \wedge \Diamond_{(0,0.04]} g_1)$	1994 / 2000
ϕ_3^{AT}	$\neg \Box_{[0,30]}(\omega \leq 4500)$	60 / 214
ϕ_3^{AT}	$\Box_{[0,10]} \neg (v \leq 80)$	1996 / 2000

B. Hoxha, H. Abbas and Georgios Fainekos, Benchmarks for Temporal Logic Requirements for Automotive Systems, ARCH 2014

Overview

- Motivation
- Preliminaries
- System Independent MITL Analysis
- System Dependent Vacuity Checking
- Experiments
- Conclusion & Future Research

Conclusions

- We developed a debugging framework for MITL
- We extended the existing LTL vacuity detection algorithms to MITL
- We used utility result to check that specification is common
- We implemented the antecedent failure detection algorithm that can find signal vacuity
- Our tool can improve the users ability to create correct MITL specifications

Future Research

- Integrate MITL analysis into ViSpec
- Finding the Coverage of specification with respect to falsifying signals
- Improving the stochastic search algorithms for falsification of the requirement in CPS with signal vacuity detection.

Acknowledgements

On-Line Survey: http://goo.gl/forms/YW0reiDtgi

Sponsors

NSF awards:
IIP-1454143
CNS-1116136
CNS-1350420
CNS-1319560

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.