A Methodology and Formalism to Handle Timing Uncertainties in Cyber-Physical Systems

by

Mohammadreza Mehrabian

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved July 2021 by the
Graduate Supervisory Committee:

ARIZONA STATE UNIVERSITY

August 2021

ABSTRACT

Uncertainty is intrinsic in Cyber-Physical Systems since they interact with human and work with both analog and digital worlds. Since even minute deviation from the real values can make catastrophe in safety critical application, considering uncertainties in CPS behavior is essential.

On the other side, time is a foundational aspect of Cyber-Physical Systems (CPS). Correct timing of system events is critical to optimize responsiveness to the environment, in terms of timeliness, accuracy, and precision in the knowledge, measurement, prediction, and control of CPS behavior.

In order to design more resilient and reliable CPS, first and foremost, there should be a way to specify the timing constraints that a constructed Cyber-Physical System must meet with considering existing uncertainties. Only then, we can seek systematic approaches to check if all timing constraints are being met, and develop correct-by-construction methodologies. In this regard, Timestamp Temporal Logic (TTL) is developed to specify the timing constraints on a distributed CPS. By TTL designers can specify the timing requirements that a CPS must satisfy in a succinct and intuitive manner and express the tolerable error as apart of language. The proposed deduction system on TTL (TTL reasoning system) gives the ability to check the consistency among expresses system specifications and simplify them to be implemented on FPGA for run-time verification.

Regarding CPS run-time verification, Timestamp-based Monitoring Approach (TMA) has been designed that can hook up to a CPS and take its timing specifications in TTL and verify if the timing constraints are being met with considering the existing uncertainties in the system. TMA does not need to compute whether the constraint is being met at each and every instance of time but it re-evaluates a constraint only when there is an event that can affect the outcome. This enables it to perform online timing monitoring of CPS for less computation and resources. Furthermore, the minimum design parameters of the timing CPS that are required to enable testing the timing of CPS are defined in this dissertation.

To my family.

ACKNOWLEDGMENTS

This dissertation was only made possible by the support and guidance of my advisor Dr. Aviral Shrivastava who his guidance and dedication inspired me over the course of my Ph.D. I would like to thank my committee members, Dr. Fengbo Ren, Dr. Hessam Sarjoughian, and Dr. Patricia Derler for their insights and feedback on my research.

TABLE OF CONTENTS

Page

LIST OF TABLES ix

LIST OF FIGURES x

CHAPTER

1 INTRODUCTION 1

1.1 Contributions of This Thesis 7

2 UNCERTAINTY MUST BE CONSIDERED IN CYBER-PHYSICAL SYSTEMS 9

2.1 Why Uncertainty Must be Considered in CPS 9

2.2 Different Uncertainty Sources 12

2.2.1 Knowledge uncertainty 13

2.2.2 Natural Variability 13

2.2.3 Decision Uncertainty 14

2.3 How to Handle Uncertainty in CPS 14

2.3.1 Conceptual Abstraction and Specification 15

2.3.2 Modeling, Simulation, and Verification 16

2.3.3 Validation and Testing 20

3 A FORMALISM FOR MONITORING TIMING SPECIFICATIONS OF CPS WITH CONSIDERING TOLERANCE 22

3.1 Signal Temporal Logic - STL 23

3.1.1 The Drawbacks of STL Statements in Monitoring 24

3.1.2 A Solution to Eliminate the Impact of Measurement Error . . 27

3.2 Timestamp Temporal Logic - TTL 29

3.2.1 The Event Representation in TTL 30

3.2.2 TTL Syntax 31

3.2.3 TTL Semantics 32
4 TTL REASONING SYSTEM 43
4.1 Background 44
4.1.1 Continuous-Time Signals 44
4.1.2 Discrete-Time Signals 45
4.1.3 Events on Signals 45
4.2 TTL Arithmetic Axioms 46
4.3 TTL Axioms and Logic Rules 46
4.3.1 TTL Axioms: 48
4.3.2 TTL Rules 49
4.4 Natural Deduction for TTL 52
4.4.1 Natural Deduction Reasoning on TTL statements 53
4.4.2 The TTL Reasoning Features 54
5 TIMESTAMP-BASED MONITORING APPROACH TO MONITOR TIMING SPECIFICATIONS OF CPS 57
5.1 TMA Algorithms for Level-based Temporal Specifications 59
5.1.1 Globally Operator 59
5.1.2 Eventually Operator 59
5.1.3 Until Operator 60
5.2 TMA Algorithms for Event-based Temporal Specifications 63
5.2.1 Latency Constraint 64
5.2.2 Simultaneity Constraint 65
5.2.3 Untimed Operators' Computation Using Timestamps 65
5.3 Time Testing Methodology 66

5.3.1 Methodology Steps . . . 67
5.3.2 Methodology Capabilities . . . 71
6 A METHOD TO QUALIFY TESTBEDS TO VERIFY TIMING BEHAVIOR OF CPS . . . 73
6.1 Distributed testbed to evaluate timing behavior of (distributed) CPS 73
6.2 Analog to Digital Converter (ADC) parameters . . . 75
6.2.1 ADC Sampling Rate . . . 75
6.2.2 ADC Resolution . . . 76
6.3 Input Impedance . . . 76
6.4 Clock Parameters . . . 77
6.4.1 Clock Fractional Frequency Offset . . . 78
6.4.2 Clock Synchronization Rate . . . 79
6.5 Testbed Capability Analysis . . . 79
7 APPLICATIONS AND EMPIRICAL EVALUATIONS . . . 82
7.1 Applications, Their Timing Specifications, and Modeling . . . 82
7.1.1 Flying Paster Application . . . 82
7.1.2 Temporal Specifications of Flying Paster . . . 84
7.2 Autonomous Intersection Manager (RIM) Application . . . 85
7.3 Applying Reasoning System on two Applications . . . 87
7.3.1 Power Grid Synchronization (Synchrophasor) . . . 89
7.3.2 Simultaneous Image Capturing for 3D Reconstruction . . . 93
7.3.3 Timing Specifications of Simultaneous Image Capturing . . . 95
7.4 Applying TTL Reasoning System on Flying Paster and RIM . . . 95
7.4.1 Consistency Checking of Temporal Specifications . . . 95

7.4.2 Consistency Checking for Autonomous Intersection Scenario 97
7.4.3 Considering Measurement Errors for Accurate Monitoring .. 99
7.4.4 Simplifying the Monitoring Logic 102
7.4.5 Reasoning about Unobservable Events 106
7.5 Using TMA for Run-time Monitoring of Simultaneous Image Capturing and Synchrophasor Applications 107
7.5.1 Monitoring Flying Paster Using TMA 107
7.5.2 Monitoring Synchrophasor Using TMA 112
7.5.3 Monitoring Simultaneous Image Capturing Using TMA 113
7.6 Using Timestamp-based Monitoring Approach (TMA) to Monitor Temporal Behavior of Globally and Flying Paster Application 114
7.6.1 Efficiency of TMA in Implementing a Single Operator 115
7.7 The Satisfaction of the Testbed Requirements 115
7.7.1 NI-cRIO Setup 116
7.7.2 ZYNQ 7000 Setup 118
7.7.3 Testbed Capability Analysis 119
REFERENCES 120
APPENDIX
A UNCERTAINTY ANALYSIS 126
B THE PROOF FOR TMA ALGORITHMS AND TTL RULES 131

LIST OF TABLES

Table Page

2.1 Different Types of Uncertainties. 10

3.1 Satisfaction Relations and Language Semantics 41

4.1 TTL Arithmetic Axioms. 47

4.2 The Introduction and Elimination Rules for TTL Timing Specifications. 49

5.1 The Rules to Create a Parse Tree from a TTL Statement 68

7.1 Seven Scenarios in Flying Paster to See the Effect of Measurement Error in the Calculation. 101

7.2 The Number of Required Lookup Tables and Flip-flops for Implementing Monitoring Tool on Zynq 7000. 106

7.3 Six Different Scenarios for Flying Paster Application. 110

7.4 Memory Requirement on FPGA for *Globally* Operator. 115

7.5 The Specifications of Monitoring Devices 116

7.6 The Specifications of Monitored CPS Applications. 117

LIST OF FIGURES

Figure Page

2.1 RSS Rules for Three Cases, i) Longitudinal Distance, ii) Lateral Distance, and iii) Merging Roads. 11

2.2 One Possible Classification for Uncertainty. Knowledge Uncertainty, Natural Variability, and Decision Uncertainty Are Three Major Sets for Explaining Uncertainties in CPS. 15

3.1 The Time Deference between Two Events Generated by Threshold Crossing. 24

3.2 The problem with monitoring STL timing statements. 26

3.3 Maximum Time Measurement Error in Signal Transitions Is δ. 28

3.4 The Process to Events from Analog Signals by Threshold Crossing. . . . 31

3.5 The Effect of Tolerance in *Latency* and *Simultaneity* Calculations. 37

5.1 a) Generating a Boolean Signal from Analog Signal, b) Calculating Until Operator in TMA. 58

5.2 FSM to Implement an *Until* Operator. 62

5.3 a) Calculating of *Latency* Constraint, b) Calculating *AND* Gate Using Timestamps. 64

5.4 a) Calculation of *Simultaneity* Constraint b) The Timed-automata to Calculate Simultaneity Constraint. 65

5.5 The FSM for Calculating AND Gate Using Timestamps. 66

5.6 Generated Parse Tree for a TTL Statement. 69

5.7 TTL Computing Blocks According to the Parse Tree in Figure 5.6. . . . 71

6.1 Time Testing Structure Diagram. 75

6.2 A Digitized Analog Signal at Two Different Sampling Rates. 76

Figure Page

6.3 An Analog Signal Sampled Using Two Different ADCs That Have the same range and different resolutions. 77
6.4 Voltage Drop on Different Signals Connected to a Resistive Load. 78
6.5 Worst-case Error between Actual Occurrence Time and Detection Time for an ADC with Sampling Frequency f_s. 80
7.1 The Schematic of Flying Paster and its Modeling Implementation. 84
7.2 Robust Intersection Manager (RIM) Algorithm. 86
7.3 Autonomous Intersection Manager Using Traxxas Cars. 87
7.4 RSS Longitudinal Distance Rule and the Relations between the Events. 87
7.5 The Schematic of Controlling Two DC Motors to Rotate in the Same Phase. 90
7.6 Modeling the Synchrophasor Using Two DC Mtors. 91
7.7 Two ArduCAM Taking Simultaneous Images of a Moving Soccer Ball. 94
7.8 The Schematic of Monitoring Simultaneous Image Capturing. 95
7.9 Latency Measurement between Contact and Cut When the Velocity of Active Roll Is Increased. Latency Measurement between Contact and Cut. w Is the Error Value between Actual Velocity and the Calculated by the Reasoning System with Considering Measurement Errors. E Demonstrates the Error Value When We Do not Consider the Measurement Errors. Ignoring the Measurement Error Can Result in an Incorrect Evaluation of the Timing Constraint and Cause a Failure. . . 102
7.10 Comparison of FF and LUT Numbers in 3 Implemented Methods. 111
7.11 #FFs Utilization in Three Methods in Different Six Scenarios in Table 7.3112
7.12 Acquired Results from Oscilloscope for the Synchrophasor. 113

Figure Page

7.13 The Results for Image Capturing Acquired from Oscilloscope. a) Images Are Taken Simultaneously. b) The Delay between the Time That the Images Are Taken Is Less Than 0.2 *s*. 114

A.1 PDF of a Gaussian distribution (continuous) and (discrete) 130

Chapter 1

INTRODUCTION

Cyber-Physical Systems (CPS) integrate physical and computational worlds to form smart, coordinated, efficient and responsive infrastructures. Deploying CPS that increase the efficiency in sectors of health, energy, aviation and freight rail by 1% will save \$186 billion in the U.S. over a 15 year period.

Time is a fundamental concept in CPS which allows the integration of discrete (cyber) and continuous (physical) domains [Shrivastava *et al.*, 2017]. CPS use sensors whose data are often time-tagged for efficient data fusion and knowledge of when the measurement was taken. Computing, communication and control commands in dynamic real-time systems need to be executed within a specified latency [Mehrabian et al., 2012]. Correct and robust orchestration of different tasks and/or distributed parts requires correct temporal behavior within and among CPS components. Current and future CPS systems such as health-care monitoring and active control devices, intelligent transportation, and electrical power systems are a few safety-critical examples requiring synchronization and latency controls.

In order to be confident about the behavior of a built or designed CPS, their timing behavior must be tested and verified. Prior to performing testing of temporal behavior, timing constraints must be expressed in a formal language. That enables robust analysis of constraint satisfiability and consistency. The formal expression enables the application developer to explicitly specify timing requirements in the design phase and provide the basis to automate the generation of application and associated test code to enable a more systematic, rigorous, and iterative verification process of the SUT (System Under Test).

Temporal logic provides the formalism to define time specifications, where evaluation of constraint satisfiability is based on reasoning about the propositions. There are several types of temporal logic which reason about variables on a discrete or continuous time domain. LTL (Linear Temporal Logic) [Bolotov et al., 2006] is defined for sequences of boolean predicates, MTL (Metric Temporal Logic) [Koymans, 1990a] is expressed on real-valued signals in discrete time and STL (Signal Temporal Logic) [Maler and Ničković, 2013a] is utilized for specifying timing constraints on real-valued signals over continuous time and more suitable for CPS.

Event-based timing constraints can be expressed in STL by using the *Rise* and *Fall* operators [Maler and Ničković, 2013a]. However, STL statements can become quite complicated, difficult to understand, and therefore, error prone. Since timing constraints are specified and written manually by humans, they should be readable and intuitive, to bridge specification at the programming language level with synthesis and validation during application compilation and verification on hardware platforms. In particular, using STL expressions to specify simple latency constraints among events become complicated, as they must be expressed in a nested manner. Furthermore, expressing the acceptable tolerance of the timing constraints make the constraint expressions more complicated.

The other parameter that should be considered in expressing the CPS temporal behavior is expressing tolerable error (i.e., Tolerance) existing in the measurement equipment in CPS manufacturing. Expressing *tolerance* is crucial to develop the right implementation of CPS and verifying their specifications since it represents the level of required precision in CPS. Tolerance values express how much degradation is acceptable for a timing specification and allows developers to know the minimum specification of a measurement system that must be used to validate the specification. In the absence of tolerance, specifications might be un-testable. For example, saying that an event must happen at $4:00$ *p.m.* is vague and infeasible to test. This is because checking whether the event happened at exactly $4:00pm$ may take unbounded time. Instead, the specification should be that a certain event must happen at $4:00$ *p.m.* ± 1 *minute*.

On the other side, since CPS are generally surrounded by a chaotic environment and they interacting with an analog world while their computation and network system are in the digital world, their behavior might be uncertain and this might impact the implemented safety requirements. In most CPS, having uncertain values is intrinsic and the developers struggle with imprecise measured values at all levels of design, modeling, simulation, verification, and validation.

Uncertainty comes from inadequate information, incorrect assumptions, erroneous data, and/or variability of natural processes. Or, the relationship among system's user, adaptation logic, and business logic might be loosely coupled. Since the uncertain values directly affect the behavior of systems, the developers should deal with them as a real challenge to guarantee that their product meet its expected timing behavior [Shrivastava et al., 2017, Chipman et al., 2015, Ma et al., 2019]. The uncertainties have a variety of sources. Drifts in measured values, aging of the physical devices, noise, and sporadic failures are some examples of the sources for the deviation. Moreover, the cyber part of the CPS discretizes continuous signals that are sampled from the physical world and this process always adds an inevitable quantization error.

The timing and the nature of inputs, the system state, and physical environment are some examples of the uncertainty sources in CPS that can be divided into two major categories for uncertainties, i) uncertainty in timings, ii) uncertainty in values. In order to develop the right verification/monitoring for a system, the effects of uncertainty on the system specification/operation should be considered [Zhang et al., 2016a] in both directions. Indeed, ignoring even minute uncertainty values, especially in the system's timings, might put vast deviations on the system behavior and affect the system's safety especially in time-sensitive safety-critical applications [Radojicic et al., 2017a, Lee, 2008, Zhang et al., 2016b]. Some real catastrophic events like the crash of LH2904 aircraft [Ladkin, 1994], Patriot Missile crash [US GAO, 1992] and Nissan's airbag issue [Charette, 2014] demonstrate that deviations in measured parameters or deflection in actuation can cause failure in a system. Although some works have been done for value uncertainty under Temporal Logic Robustness domain (e.g., robustness on MTL [Annpureddy *et al.*, 2011] and/or STL [Annpureddy *et al.*, 2011]), considering timing uncertainties in run-time verification is a serious need.

This work focuses on temporal monitoring of Cyber-Physical Systems with considering tolerance as a part of high-level specification and uncertainty in low-level implementation. Therefore, in chapter 2 the source of uncertainties and the way to handle them in different steps of CPS developments are introduced.

Timestamp Temporal Logic (TTL) [Mehrabian and other, 2017] is a temporal logic language expresses the timing constraints among a set of events with having the ability to have time tolerance value as a part of language. It is a formalism for expressing temporal specifications of CPS that receives the events on signals and evaluate the timing specifications by doing the calculations on the event's timestamps. TTL proposes 5 event-based temporal operators for expressing the timing specifications of CPS and consider a room in each operator to add the maximum tolerable error in evaluating the timing predicates.

Indeed, CPS struggle with a sort of uncertainties while the measurement in cyber side also is not free of errors. One solution for such erroneous environment is considering the measurement uncertainty in the calculation. Therefore, it is beneficial if the value of uncertainty comes to the formalism and has its own required role in calculation while the temporal logic statement is evaluated. By adding the uncertainty to the evaluation of TTL predicates, the system designers develop a realistic system descriptions where the monitoring developers are able to monitor the same specifications with considering the same existing uncertainties. In addition to considering the measurement uncertainty in the logical predicates, TTL reduces the number of temporal operators and consequently diminishes the required space circuit for run-time monitoring. The events are expressed as direct single predicates using threshold crossing of the signals and it is not needed to generate the event using the STL operators that causes the statements long and hard to read and implement. Furthermore, a methodology comes along with TTL formalism making the process of monitoring more systematic especially for nested and complicated temporal statements. The proposed methodology, introduced in chapter 3, gets joined and nested temporal statements, parses them in a parsing graph, make the corresponding block diagram and finally, implement each block in TMA. The methodology manages the input/outputs of the blocks to produce the right verdict for the input temporal statements.

Developing the monitoring approach to verify the CPS temporal specifications can be more systematic and efficient. It is shown in chapter 4 that each TTL operator can be translated into mathematical predicate using the Mathematical Logic formalism. As a result, it becomes possible to examine a set of TTL statement by converting them into mathematical predicates. By hiring the rules in Mathematical Logic, a developer is able to do some extra operations on the predicates. Therefore, in some cases the possibility of inconsistency between the statements can be checked or it is possible to combine several TTL statements to archive a higher degree of efficiency for the size of the monitoring circuit in online method. Such capabilities become practical if we define the required axiom and logical rules for each TTL operator. Using the traditional natural deduction model is the key to define the rules and the way to deduce new statements from some TTL hypothesises and predicates. The benefit of the natural deduction on TTL is having the ability to take both *user-defined tolerance* as well as system *uncertainties* into account when new statements are deduced. Chapter 4 summarizes the axioms and proposes the rules and the style of proof in reasoning process for TTL.

As an efficient monitoring technique, Timestamp-based Monitoring Approach (TMA) is presented in the next part. TMA utilizes timestamps of changing values on signals to evaluate the required temporal specifications in CPS. TMA is able to monitor the generic temporal operators of STL (i.e. *Globally*, *Eventually*, and *Until*) using the timestamps of the transitions from $\perp$ to $\top$ and vice versa. This technique, presented in chapter 5, stores just last two timestamps of events on signals and by running its algorithms can return the right verdict about satisfactory of a timing constraint with its specified time *tolerance* and existing *uncertainty* in the measurement system. The proposed method has the capability to be implemented for either online and offline monitoring of CPS. In online monitoring, the required space on an FPGA board is constant and not related to the horizon of the temporal operator. In fact, TMA forms a Timed Automaton for each operand and implement the generated automota on FPGA.

Considering user-defined *tolerance* value and existing measurement uncertainties in converting the continuous world (Physical) to discrete (cyber) has another benefit by which the suitable equipment for monitoring can be chosen. Indeed, an arbitrary monitoring device does not necessarily have enough precision for run-time monitoring of a specific CPS. The data acquisition modules, the frequency of the equipment's clock, the synchronization frequency and its maximum drift in distributed systems, the resolution of the Analog to Digital converters and the other parameters affecting the system timing should be considered and pass a certain minimum qualification exam. Therefore, by expressing the CPS timing constraints in TTL where the operators contain the maximum acceptable and accumulated errors as ε, the developers determine the minimum requirements for the monitoring equipment. Therefore, if the maximum error in the measurement part of monitoring equipment is small enough comparing with ε it is qualified for the run-time verification of that specific CPS. Regarding this capability of TTL, a set of time-related parameters are defined in chapter 6 to standardize the monitoring equipment. By measuring those parameters, the maximum timing error for the monitoring device can be calculated and therefore, its ability to monitor a certain CPS is examined.

Finally, in chapter 7 four time-sensitive CPS applications are introduced. In this chapter, the effectiveness of the proposed methods on real applications are examined.

1.1 Contributions of This Thesis

The contributions of this thesis can be summarized as blow:

- A temporal logic formalism, Timestamp Temporal Logic (TTL) is presented that it helps to have a more efficient and accurate monitoring. TTL provides the required potential to have efficient monitoring process by proposing succinct and intuitive temporal operators. By TTL, it is possible to have precise and accurate run-time verification process since it considers the user-defined time tolerance and as a part of the language and takes measurement errors into account as uncertainties in evaluations of temporal expressions.

- A reasoning system by applying natural deduction proof approach on mathematical logic is proposed to prepare the logical statements for monitoring and/or reduce the number of predicates. By doing the consistency check and combining the logical statements, the run-time verification will be more meaningful and testing developers can provide a shrunk circuit for FPGA. Moreover, the proposed axiom and rules consider tolerance and uncertainties in all mathematical calculations.

- An efficient online monitoring method is presented that implements the run-time verification of CPS temporal specifications efficiently in terms of the required area of FPGA. TMA does not need to store the value of signals based on their related interval and just do the calculations based on very last timestamps. Moreover, since TMA utilizes TTL, it is able to do the right monitoring because TTL-based calculations have the required parameters for tolerances and uncertainties.

- In order to qualify a testbed to monitor temporal specifications of CPS, the parameters having significant impacts on the precision of testbed in measuring time are introduced (i.e., some metrics are proposed). By knowing the maximum allowed tolerable error in TTL statement, the accurate-enough equipment based on proposed metrics are chosen for run-time verification. Therefore, the way to calculate the total error and how it can be covered in TTL statements are studied.

- Four different time-sensitive applications have been implemented. Their proper functionalities have been examined by TMA and the TTL methodology/reasoning where the precision of the testbeds determined by the proposed standard.

Chapter 2

UNCERTAINTY MUST BE CONSIDERED IN CYBER-PHYSICAL SYSTEMS

Outcomes or events that cannot be predicted with certainty are often called risky or uncertain values. In fact, there is a distinct difference between *Uncertainty* and *Risk*. Risk describes situation for which probabilities are available and the result has a negative effect. It is used to describe the likelihood of various events, the system state and/or outputs. When the ranges of possible events are known and their probabilities are measurable, risk is called objective risk. If the probabilities are based solely on human judgement, the risk is called subjective risk. On the other side, if the probabilities of a set of events/outputs/states cannot be quantified or the occurrence of events are unpredictable, the problem is a sort of *uncertainty* and not risk.

2.1 Why Uncertainty Must be Considered in CPS

It could be said that the presence of uncertainty is the only thing that is certain in a Cyber Physical Systems (CPS). Uncertainty in information is inherent in future oriented planning efforts. It comes from inadequate information and incorrect assumptions, as well as from the variability of natural processes.

Sensors have noise so even if they sit next to each other the sampled values may not be the same. One can see that it is not a single source that causes uncertainty in CPS, rather it is many different sources of uncertainty sometimes compounding each other [Esfahani and Malek, 2013, van de Lindt et al., 2018]. The source of uncertainties are summarized in Table 2.1. In fact, the table demonstrates different sources fro uncertainty and their impacts on CPS operation with giving an example

Table 2.1: Different Types of Uncertainties.

Uncertainty	Definition	Impact on CPS	Example in CAV's domain	KU [a]	NV [b]	DU [c]
Physical Process	Uncertainty in the characteristics of a physical process.	Malfunctioning about the delivered service.	Miscalculation for expansion of metal in different temperatures in a wheel's shaft which causes wrong steering.	🟩		
Sensor (Measurement)	Measuring quantitative values with the presence of disturbances (noises).	Problems with understanding the environment.	The noises at the frequency of ultrasonic sensor that causes side accidents.		🟩	🟩
Actuator	Problems in delivering the right services at the right time.	Malfunctioning about the delivered service.	Failing to active brake system in-time.		🟩	🟩
Effector	Uncertainty about the future state of a robot.	Impacts the coordinated operations.	Wrong path planning when a CAV approaches a merging point.	🟩		🟩
Controller	Uncertainty in behavior, execution time, and context of a controller.	Malfunctioning about the delivered service.	Failing to achieve supposed velocity in an interval that increases the chance of collision.	🟩		🟩
Instrumental	Uncertainties caused by instrumentation among different parts.	Malfunctioning about the delivered service.	The connectivity among electrical parts changes the voltage of the connected wires.	🟩		🟩
Network	Lack of information about network at run-time.	Impacts the coordinated operations.	late package for Intersection Manager or the other CAVs causes late decision in one CAV and increases the chance of collision.	🟩	🟩	
Time	Lack of information about the occurrence of event or duration of an activity.	Malfunctioning about the delivered service.	The late/soon event generation causes late/soon actuation (e.g., in braking system) and increases the chance of collision.	🟩		🟩

[a]Knowledge Uncertainty
[b]Natural Valriability
[c]Decision Uncertainty

in Connected Autonomous Vehicle's domain. If these sources of uncertain are not dealt with small errors can eventually compound into very significant errors. In fields like automation, manufacturing, and autonomous vehicles not dealing with this uncertainty correctly can be catastrophic even resulting in death.

CPS systems are automated and therefore uncertainty must be taken into account when programming them or it will not be dealt with. In order to deal with the error, one must first understand it. To that end there is a significant amount of work in classifying and modeling uncertainty. These methods range from extensive testing such as hardware in the loop (HiL) to software modeling using simulation. In the end the goal is the same, understand and quantify uncertainty so that it can be dealt with in all the situations the CPS will encounter [Esfahani and Malek, 2013]. Sometime this is still not possible and the system must be set up to gracefully degrade (fail) in a situation that it cannot understand. Uncertainty is inevitable, however through correct classification, modeling, and creation of methods to deal with it the risks can be mitigated.

In order to see the impact of uncertainties on the CPS operation, an example is given here in the domain of autonomous vehicles. The Responsibility-Sensitive Safety (RSS) [Shalev-Shwartz et al., 2017, Khayatian et al., 2021] provides a framework to determine the safe distance (with considering the worst-case scenarios) to avoid accidents on the roads. The method proposes a set of rules to avoid collisions and in a case of accident, it has a certain methodology to put the blame on the agent which could not follow the rules. Figure 2.1 depicts three possible scenarios. There are longitudinal distance between the blue (C_f) and green (C_r) cars, lateral distance between the green (C_1) and red (C_2) cars, and a merging situation between the red (C_m) and yellow (C_j) cars. For longitudinal case if the longitudinal distance becomes less than d_{min}^{long}, C_r should react in time. Similarly, in the lateral case if the lateral distance becomes less than d_{min}^{lat} both C_1 and C_2 should react in-time. In the last case where C_j approaches the merging point, the closest vehicle to the merging point has the right of the road ($min(d_m, d_j)$) and the second vehicle just maintains its longitudinal distance to it (d_{min}^{long}).

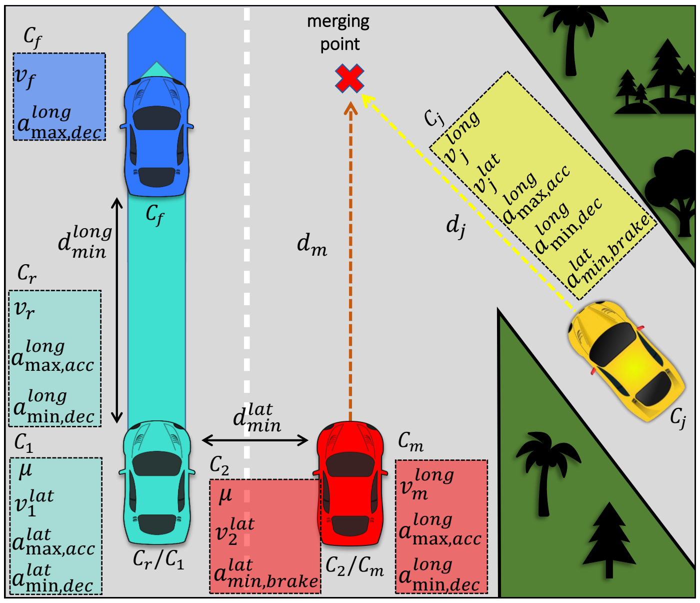


Figure 2.1: RSS Rules for Three Cases, i) Longitudinal Distance, ii) Lateral Distance, and iii) Merging Roads.

For instance, the equation below shows the way to calculate d_{min}^{long} for C_f and C_r:

$$d_{\min}^{long} = \Big(v_r\rho + \frac{1}{2}a_{\max,\text{acc}}^{long}\rho^2 + \frac{(v_r + \rho a_{\max,\text{acc}}^{long})^2}{2a_{\min,\text{dec}}^{long}} - \frac{v_f^2}{2a_{\max,\text{dec}}^{long}} \Big) \tag{2.1}$$

where v_r and v_f are velocities of the rear and front vehicles, $a_{max,acc}^{long}$ is the maximum acceleration rate and, $a_{min,dec}^{long}$ and $a_{max,dec}^{long}$ are the minimum and maximum deceleration rates respectively. As a fact, the measured parameters in the field at run-time are v_r, v_f, and ρ while the other parameters (e.g. $a_{max,acc}^{long}$, $a_{min,dec}^{long}$, and $a_{max,dec}^{long}$) are specific and predefined for each vehicle. In order to show the impact of existing uncertainties on the safety of such a system, we can do the calculation in two different fashions: i) without considering and ii) with considering uncertainties in measured values. Hence, the equation 2.1 is used for the former case and for the latter case it should be changed a little bit with considering the three following parameters: v_r, v_f, and ρ where they are replaced with v_r', v_f', and ρ' as $v_r' = v_r \pm \Delta v_r$, $v_f' = v_f \pm \Delta v_f$, and $\rho' = \rho \pm \Delta\rho$.

The above formulas show that each parameter does not have an exact value, but in a range. Therefore, if we measure them as $v_r, v_f = 20.0m/s$ and $\rho = 1.0s$, and $\Delta v_r, \Delta v_f = 1m/s$, and $\Delta\rho = 0.2s$ they can be $19.0 \leq v_r, v_f \leq 21.0$, and $0.8 \leq \rho \leq 1.2$. If we substitute the exact values in equation 2.1, the calculated $d_{min}^{long} = 210m$. On the other side, with considering the ranges and assuming the possible worst-case scenario, the new minimum longitudinal distance is $d_{min}'^{long} = 60m$. Obviously, having such big differences between real values and calculated ones puts the entire system in an unsafe situation.

2.2 Different Uncertainty Sources

A CPS at it heart is a digital device interacting with the physical world and this interaction is the source of the uncertainty. Sensors, actuators, communication, timing, temperature fluctuations, operating systems, hardware faults, software faults, vibration, clock skew, and far more than can be listed here are sources of uncertainty. In this section, the different types of uncertainties, their sources, and impacts on CPS operation by which developers get the abilities to to handle existing uncertainties during their design and verification/validation processes are introduced.

2.2.1 Knowledge uncertainty

The lack of knowledge where it is not possible to exactly describe the current state, a future result, or more than one possible outcome [Cailliau and Van Lamsweerde, 2015]. As Figure 2.2 depicts knowledge uncertainty includes parameter value uncertainties and model structure where they are about the uncertainty in system parameters and its model.

Epistemic uncertainty

Epistemic uncertainty potentially comes from the lack of knowledge when we model a system. Unlike *Aleatory Uncertainty* [1] , Epistemic uncertainty is reducible. Since it is caused by a lack of information, it can be resolved by gathering complete information for related models. There are some mathematical solutions that can be used to model epistemic uncertainty, such as Bayesian estimation [Oberkampf et al., 2002].

2.2.2 Natural Variability

The time-series values usually shows the historical conditions including droughts and wet periods [Ghanem and Wojtkiewicz, 2004], and they are often actual, or at least based on, historical data. One thing that is clear about natural uncertainty is that natural variability cannot be reduced by improving the model's structures,

[1] That is explained in the next section

better calibration of model parameters, or even increasing the resolution of simulations [Ciffroy and Benedetti, 2018, Ciffroy, 2020].

Aleatory uncertainty

Aleatory uncertainty is the inherent uncertainty introduced by the physical part of a system [Oberkampf et al., 2002] and is a kind of natural variability. This type of uncertainty cannot be reduced since it exists in all variables from a physical environment. This is an impact of natural variability and hence, parameters can be typically modelled as a probability distribution.

2.2.3 Decision Uncertainty

Uncertainty in system modelling can result from unanticipated changes in modelled systems. The changes can include deviations in nature, defined human objectives/activities/interests, demands and their impacts [Davis and Hall, 2003]. Comparing field data with modelled data in the calibration process can yield incorrect calibrations if operating policies actually implemented in the manufactured system differ significantly from those built in models. For instance, it is needed to know what operators react in presence of stress and the response to this question helps to realize uncertainties of an operator.

Based on the definitions for different types of uncertainties, each of listed uncertainty types in Table 2.1 can be categorized in one category.

2.3 How to Handle Uncertainty in CPS

Providing the right behavior for CPS is challenging and there are a lot of approaches to make them correctly according their required specifications. One solution is bottom-up approach where deliver precision timings and functionalities. However,

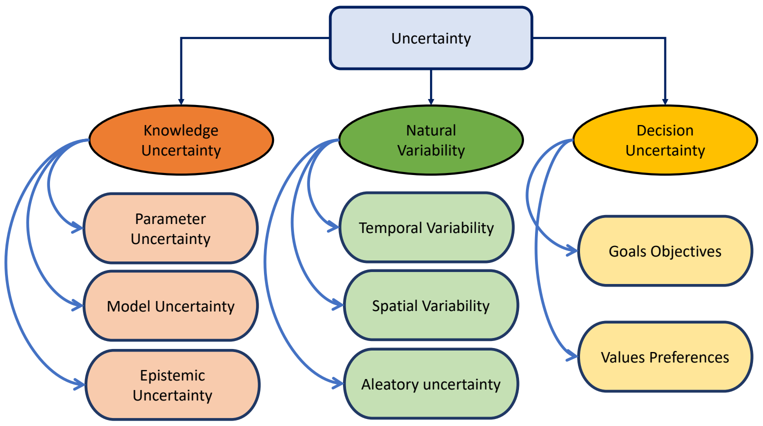


Figure 2.2: One Possible Classification for Uncertainty. Knowledge Uncertainty, Natural Variability, and Decision Uncertainty Are Three Major Sets for Explaining Uncertainties in CPS.

it leaves some questions unanswered about the software for the design the programming languages, and the methodologies to make them correctly [Edwards and Lee, 2007]. On the other side, top-down techniques need model-based design where *programs* are replaced by *modules* representing system behaviors of interest [Sztipanovits and Karsai, 1997]. Finally, the software comes from the modules. The attractive part of model-based approach is its rich possibilities for interfacing specifications and compositions. This section talks about the levels of developing CPS, containing i) *Expressing System Specifications*, ii) *Modeling and Simulation*, and iii) *System Validation*, in Model-based approach and the requirements that are needed in each level for verification and validation.

2.3.1 Conceptual Abstraction and Specification

Conceptual abstraction is responsible for providing the design-characteristic facets of CPS regarding hierarchy parameters and system modularity [Hehenberger et al., 2016]. CPS designers start this phase with a definition of a system as an abstract representation and they decide what system aspects (structures or behaviors) of systems to be included and what others to be not included. One example of conceptual model is the Apollo's architecture model in [Git, 2021] where the software view shows the conceptual position of each module in a CAV and the rational relationships among all high-level modules [2] . In the conceptual design the uncertain parameters should be defined.

2.3.2 Modeling, Simulation, and Verification

As a part of CPS implementation, system modeling comprises a set-up of cross-multidiscipline models to determine the functional properties and system's parameters. In the simulation and verification are two more steps to run a mimic of the system in modeling software such as MATLAB. In all three processes we need to consider uncertainties.

System Modeling

The target of having system modelling phase is to propose a cross-multidiscipline model to determine the required functional, temporal, and system parameters. In this area, there are major three concerns; i) how to model different components from different disciplines, ii) how to model the interface between each pair of components, and iii) how to integrate modelled components [Hehenberger et al., 2016]. If one needs to cover uncertainties not just for modeling but also for verification, s/he has to conduct systematic approach to cover all those three steps. In deed, modeling CPS requires considering various aspects of CPS such as cyber and physical parts, dynamic behavior, involved both functional non-functional properties. To consider these requirements, several researchers have used and proposed some modeling languages. In the modeling scope, the uncertainty sources, the quantization techniques and the integration of different uncertain values should be defined carefully (some

[2]The modules are Perception, Prediction, Planning, Localization, Control, etc.

examples for quantification methods are in Appendix A).

Additionally, one can consider uncertainties in perspective of timed-automata. To be specific, uncertainty will be in parameters (variables) when CPS consists of a single state (in real-world, a state is a process). For example, when calculating R.S.S. rule's safety distance, driver's response time may differ by vehicle. On the other side, uncertainty will be in not only parameters but also in states and transitions if CPS consists of multiple states (processes) and transitions. For instance, even though the rear car, in longitudinal distance scenario, decided to brake to maintain safety distance, the vehicle might fail to decelerate because of malfunctioning. This means transitions among states might be non-deterministic so that there are multiple states and transitions for specific state and each transition follows probability. Furthermore, since the verification process definitely requires a literature of temporal logic, uncertainties have to be expressed in this way combined with target CPS.

Therefore, in this stage of designing, we should consider uncertainty in the following domains:

- **Uncertainty in parameter**: Parameters can be exact, arbitrarily fixed, stochastic (samples provided), or even uncertain (no sample).
- **Uncertainty in state**: State can be constant (discrete state) or changing every moment (continuous state). Unknown state can also exist.
- **Uncertainty in transition**: Transition can be deterministic or non-deterministic (transition follows probability). Unknown transition can also exist.

System Simulation

Simulation of a system provides a virtual environment supporting development, testing, and analyzing CPS regarding the designer's modeling and specifications. The simulation of CPS generally comprises creation and simulation of hybrid models containing multiple levels of abstraction along with combining different formalisms [Mustafiz et al., 2016]. The literature proposed different types of simulation as below:

- Continuous, the model is formulated as a sort of differential equations;
- Discrete event, the model involves objects and the relationships among objects;
- Hybrid, an analytical submodel within a discrete event model;
- Monte Carlo, the stochastic process to solve deterministic problems;

while in the CPS domain we mostly need the hybrid simulation [Tolk et al., 2018] and uncertainty-wise simulation should be a part of analyzing. The example for RSS domain is having the Ordinary Differential Equation (ODE) to control the velocity (continuous domain) in rear vehicle and the braking system to keep the safe distance to the front car (discrete domain). In both system, we should consider uncertainties.

Over-approximation, Shanon expansion, and Affine Arithmetic Decision Diagrams (AADD) [Radojicic et al., 2017b], are some examples for simulation to consider uncertainties in hybrid systems.

Verification

In the context of interacting systems, software and/or hardware systems, the formalism *Temporal Logic* is commonly used to specify the timing requirements of CPS and utilized in both sides of system design and verification. The temporal logic languages like LTL (Linear Temporal Logic) [Pnueli, 1977], MTL (Metric Temporal Logic) [Koymans, 1990b], STL (Signal Temporal Logic) [Maler and Ničković, 2013b] can specify the system behavior containing the sequences of states and/or events, or the output values. Such specifications can be translated into monitoring programs which observe the system operation and verify the required specifications. In order to have the right implementation for CPS and its verification process, we need to take uncertainties into account with following point of views.

-Property formalization should express Tolerance:
In the domain of properties for temporal constraints, one of the challenges of existing temporal logics, as languages to express the CPS's properties, is that they do not specify the tolerance with which a timing constraint must be met. In absence of tolerance, we must assume that the temporal logic formalisms express the timing constraints with infinite precision. For instance, in developing seat belt and airbag in a vehicle, once the vehicle hits an obstacle, the time between releasing the seat belt and opening airbags should not be less or more than a special value (e.g. 30 ms) to save the passenger from internal organ injury. The expression in English is "`after a rapid deceleration, the time between seat belt pretensioner (event A that happens before), and airbag deployment (event B) should be 30 ms`. However, since time is a continuous variable, it is impossible for the delay between the two events to be exactly $30ms$. A time-sensitive application cannot be implemented, designed or tested with infinite precision. In this example, it is not clear that if the delay is 30.00001 is acceptable or 30.000000001. Or, even none is the right answer. If both are incorrect, without knowing the acceptable number of digits after the floating point position, the comparison would be limitless [Air, 2021].

Hence, as a fact, expressing timing specifications without considering tolerance is meaningless. This problem becomes more serious when we know that the measuring system is not perfect and always has errors [3] . Therefore, it is needed to have the tolerable error as a part of specification and hence, mentioned in the formal temporal constraints.

[3]e.g., due to ADC and quantification conversions.

-Verification method should guarantee the its correctness in the presence of uncertainties:
In CPS verification, it is needed to take all uncertain values into account with considering worst-case scenarios. However, some satisfactions might fall in the tolerable duration category which referred as the *gray area*. It means, it is not clear that the specified tolerance is enough to cover the overall uncertainty for a certain property. If a timing constraint is met within the gray area, the rules must detect them as violated since it may be evaluated as met just because of uncertainty in the measurement system. Based on such rule, some detected timing violations will be false positive cases (when a violation is reported while it is met) but there will be no false negatives. Therefore, from this point of view, the monitoring system should provide a guarantee to detect all violation even the ones in the gray area and this become possible with doing the monitoring process in a conservative way.

2.3.3 Validation and Testing

CPS Testing and Validation is required to ensure that they operate with high reliability and can operate as safety-critical applications. Since monitoring systems validate manufactured CPS interacting with uncertain world, having them becomes important because they can see the impact of uncertain values on the final systems' behavior. In this regard, two criteria should be met: i) the testing algorithms must take the uncertain values into account, and ii) the testing equipment should be qualified regarding the monitored systems.

In the verification process, testing system is connected to the System Under Test (SUT) and it is like a black box while the output signals are read to verify/validate the operation. Regarding reading the signals, the monitoring system deals with values. Hence, considering uncertainties explained in section 2.2 should be a part of monitoring implementation as well to make sure there is no discrepancies between the designed systems from one side and verified/validated system from the other side. As an example, when RSS minimum longitudinal distance in rear vehicle is calculated, the run-time verification system that is connected to the CPS and CPS itself should have a unique understandings about uncertainties. Otherwise, the judgment made by the validation system is not correct. For example, the way to convert the analog signals to digital and how to consider the measurement error cased by the conversion should be taken into account should be similar in both sides.

The other concern is having the right and quantified equipment for validation. For instance, a device with millisecond precision is not qualified to monitor a cyber-physical system that its maximum tolerable error is in the range of microsecond.

Chapter 3

A FORMALISM FOR MONITORING TIMING SPECIFICATIONS OF CPS WITH CONSIDERING TOLERANCE

In order to test the performance and verify the correctness of Cyber-Physical Systems (CPS), the timing constraints of the system as its behavior must be met. There have been many efforts to verify and ensure CPS reliability. Formal methods, simulation, and testing have been applied to increase the quality of parts of CPS but it is still a challenge to formally verify complete CPS [Zheng and Julien, 2015]. Therefore, the second option is to execute the CPS and detect anomalies at run-time. In order to monitor the timing specifications of CPS, they should firstly, expressed in a formal language like temporal logic. There are several temporal logic languages to express the timing specifications of CPS. CTL, LTL, and MTL are some examples while Signal Temporal Logic (STL) is a fit for CPS since it can express the timing specifications for analog signals over continuous time. STL can efficiently and succinctly capture the timing constraints of a given system model. However, many timing constraints on CPS are more naturally expressed in terms of events on signals.

While it is possible to specify event-based timing constraints in STL, such statements can quickly become long and arcane in even simple systems. Timing constraints for CPS, which can be large and complex systems, are often associated with tolerances, the expression of which can make the timing constraints even more cumbersome using STL. The other and more important issue is about the problems in monitoring of STL statements. Since the cyber side in CPS is discrete and quantized it makes inaccuracy in STL monitoring. The details of such problems are coming sections.

3.1 Signal Temporal Logic - STL

Before start to discuss the STL problems, the logic should be introduced.

The basic formula of $STL_{[a,b]}$ are defined by grammar

$$\psi := p \mid \neg\psi \mid \psi_1 \vee \psi_2 \mid \psi_1 \mathcal{U}_{[a,b]} \psi_2$$

where p belongs to a set $P = \{p_1, p_2, ..., p_n\}$ of propositions. From basic $STL_{[a,b]}$ operators, one can derive other Boolean and temporal operators, *Globally* and *Eventually* as below:

$$\Diamond_{[a,b]}\psi = \top \mathcal{U}_{[a,b]}\psi$$

and

$$\Box_{[a,b]}\psi = \neg\Diamond_{[a,b]}\neg\psi$$

$STL_{[a,b]}$ formula are interpreted over n-dimentional Boolean signals that are generated by crossing real-value signal with real numbers. The satisfaction relation $(s,t) \models \psi$ indicating that signal s satisfies ψ starting from position t, is defined inductively as follows:

$(s,t) \models p \leftrightarrow \pi_p(s)[t] = \top$

$(s,t) \models \neg\psi \leftrightarrow (s,t) \not\models \psi$

$(s,t) \models \psi_1 \vee \psi_2 \leftrightarrow (s,t) \models \psi_1 \; or \; (s,t) \models \psi_2$

$(s,t) \models \psi_1 \mathcal{U} \psi_2 \leftrightarrow \exists t' \in [t+a, t+b] \; s.t.(s,t') \models \psi_2 \; and \; \forall t'' \in [t,t'], (s,t'') \models \psi_1$

Eventually and *Globally* operators can be define as:

$(s,t) \models \Diamond_{[a,b]}\psi \leftrightarrow \exists t' \in [t+a, t+b] \; s.t.(s,t') \models \psi$

$(s,t) \models \Box_{[a,b]}\psi \leftrightarrow \forall t' \in [t+a, t+b] \; s.t.(s,t') \models \psi$

In brief, we evaluate the above operators for the next $[a, b]$ interval at time t (the current time). $\Diamond_{[a,b]}\psi$ means ψ should become $\top$ anywhere in $[t+a, t+b]$, $\Box_{[a,b]}\psi$ means ψ should be always $\top$ in $[t+a, t+b]$, and $\psi_1 \mathcal{U} \psi_2$ means ψ_2 should become $\top$ (e.g. at t') in $[t+a, t+b]$ and ψ_1 should be constantly $\top$ from now (t) to t'.

3.1.1 The Drawbacks of STL Statements in Monitoring

In monitoring process, the system description is represented in one or the combination of the above formula as **STL statements** and they should be implemented in software or hardware codes to check those specifications.

One of the shortcomings of STL is about the expressions when there are combined or complex statements. One example is when we represent the time difference between two events. In figure 3.1, there are two analog signals, $s_1(t)$ and $s_2(t)$. One possible timing constraint on these two signals is the time deference between the events generated by threshold crossing. For instance, the time difference between $s_1(t)$ when it crosses th_1 from above to the time at which $s_2(t)$ crosses th_2 from below should be less than 2 seconds.

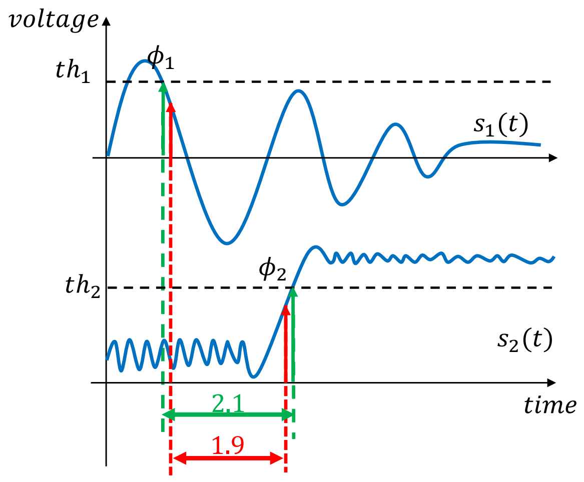


Figure 3.1: The Time Deference between Two Events Generated by Threshold Crossing.

STL has the capability to express the events using Future and Past operators, Until ($\mathcal{U}$) and Since (S) [1] . The expressions below show how to express rising edge $\uparrow \phi$ and falling edge $\downarrow \phi$ [2] events on Boolean signal ψ in STL.

$$\uparrow \phi = (\psi \wedge (\neg\psi S \top)) \vee (\neg\psi \wedge (\psi \mathcal{U} \top))$$

$$\downarrow \phi = (\neg\psi \wedge (\psi S \top)) \vee (\psi \wedge (\neg\psi \mathcal{U} \top))$$

By these definitions, it is possible to have such timing constraint between two events:

$$\Box(\downarrow \phi_1 \rightarrow \diamond_{[0,2)} \uparrow \psi_2)$$

when $\psi_1 = s_1(t) < th_1$ and $\psi_2 = s_2(t) > th_2$.

By substituting $\downarrow \phi_1$ and $\uparrow \phi_2$ with their STL equivalent expressions, we can represent the time difference between ϕ_1 and ϕ_2 events. However, expressing temporal specifications have the potential to become complex when the events are the concern even in very simple timing constraints. As it can be seen in this example with just two events, by substituting the event expressions in STL statement, it becomes long and complex [3] . Since, timing specifications are mostly read, written, analyzed, and implemented by human, they are error prone and complicated ones may be implemented in monitoring process with errors.

The other more important issue is the problem with monitoring of timing specifications expressed in STL. As a fact, monitoring systems cannot correctly evaluate

[1] S is similar to *Until* but for past. $(s,t) \models \psi_1 S \psi_2 \leftrightarrow \exists t' \in [t-a, t-b]\ s.t. (s,t') \models \psi_2\ and\ \forall t'' \in [t,t'], (s,t'') \models \psi_1$.

[2] The rising edge is threshold crossing from below and falling is from above.

[3] It can be worse in the constraints that have more events involved.

STL statements. This is because the real-time signals should be discretized for processing in the cyber side. The uniform discretization process always struggles with error since no measurement device is perfect. There is an example in figure 3.2 where the STL statement is $\Box_{[1,5]}\psi$ and a monitoring device monitors signal ψ to evaluate the statement. In this example assume the sampling time is $0.2s$ ($\delta = 0.2s$) and the sampling is started from time zero. Therefore, the sampled times (time steps) are $\{0, 0.2, 0.4, ...\}$ and there is a measurement error of 0.2 [4] .

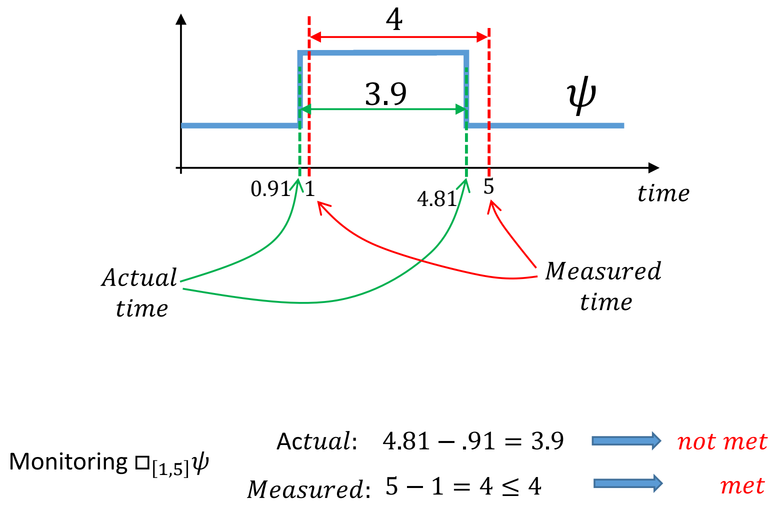


Figure 3.2: The problem with monitoring STL timing statements.

As the statement $\Box_{[1,5]}$ defines, in the example, the evaluation for the current time t is true if in the next 1 to 5 seconds (i.e. $t+1$ to $t+5$) the signal value is true continuously. In the other words, the width of positive pulse should be at least $5 - 1 = 4$ to have the evaluation of true for now. As figure 3.2 depicts, ψ is true at real-time $0.91s$ and becomes false again at $4.81s$. If we do the calculation to know the satisfaction for $\Box_{[1,5]}\psi$ in real time, the width of the positive pulse

[4]In uniform discretization process, the threshold crossing is rounded to the upper number. For Example, if there are threshold crossings for 0.09 or 0.19, both are converted to 0.2 in discretization.

is $4.81s - 0.91s = 3.9s$ showing the temporal specification has not been satisfied. However, since in the real implementation of the monitoring device everything is discreted, the detected values for the rising and falling edges are $1s$ and $5s$ instead of $0.91s$ and $4.81s$ respectively. If we do the same calculation, the result shows that $\Box_{[1,5]}\psi$ is met for now while it is true not in real.

This false positive evaluation can make serious problems particularly in safety-critical applications because there is a chance in which the monitoring device shows the critical timing constraint is met while it is not met really. Assume, there are some timing constraints in a vehicle for the braking system and they are violated in test drive and the run-time verification system shows they are satisfied. Clearly, such wrong evaluation can cause a catastrophe.

This problem is real because even with less value for measurement error for the time the events (δ) there exist a level of inaccuracy and a very slight error can cause a catastrophe in time-sensitive safety-critical applications.

3.1.2 A Solution to Eliminate the Impact of Measurement Error

As a solution to fix this problem of STL monitoring, we can easily consider the time measurement error in the system specification. It means we should express timing specifications with considering the maximum possible measurement error. In order to show the way to consider such error, let's look at *Globally* statement again.

As figure 3.3 depicts, we know a rising edge in Boolean signal ψ can occur anywhere between $n\delta$ and $(n+1)\delta$, and the measurement system detects it at time step $(n+1)\delta$ because it rounds the values to the upper number. Therefore, the maximum possible error in the right side is δ. If we express the left side for the interval by "$(\frac{a}{\delta}\delta - \delta$" [5] , we are saying that any time from a little bit after "$a - \delta$" until the next time

[5]Which is open interval for the left.

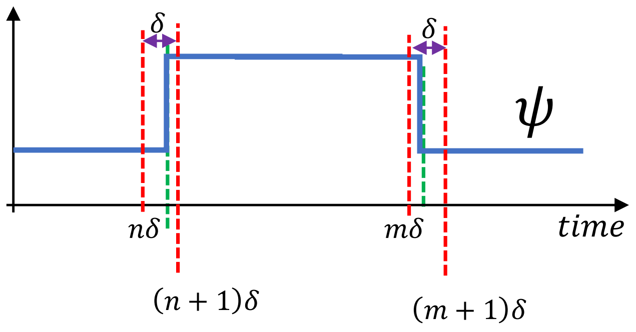


Figure 3.3: Maximum Time Measurement Error in Signal Transitions Is δ.

step is accepted. We should not take the point "$\frac{a}{\delta}\delta - \delta$" as a part of the interval because it belongs to the last sampling time. For the right side of the interval, we should consider "$\frac{b}{\delta}\delta]$" and it is closed on the right because in this sampling time, the calculated time step is rounded to the right (all values between $m\delta$ and $(m+1)\delta$ are rounded to $(m+1)\delta$) and, therefore "$\frac{b}{\delta}\delta$" itself is a part of discretized number. Hence, the modified interval with considering the measurement error is $(\frac{a}{\delta}\delta - \delta, \frac{b}{\delta}\delta]$. For instance, in the example in figure 3.2, the interval in $\Box_{[1,5]}\psi$ should be converted to $(0.8, 5]$. By the modified interval, the width of the positive pulse should be more than $5s - 0.8s = 4.2s$ to the statement is evaluated to true. In this new condition, the actual width is $4.81s - 0.91s = 3.9s$ that shows the constraint is not actually met $(3.9s < 4.2s)$. In the measuring device, we have $5s - 1s = 4s < 4.2s$ which demonstrates that the monitoring device evaluates the timing constraint correctly and shows it is not met.

As it is showed, by modifying the constraint interval, actually the constraint redefined conservatively. In the other words, it guarantees that if the timing requirement is not met in the device under test, the monitoring says it is not met as well. Such guarantee is one-sided because it is conservative in one side to show it is not met when it is really not met. In a case when the constraint is met, the monitoring system may say the constraint is not met. The latter case dose not cause serious problem since it just issues false alarms. The false alarms are the cost we pay to be aware of the accurate timing violations by the monitoring system. As a fact, we cannot cover both sides since they contradict. However, it is required to cover false positive since it is really needed in monitoring of safety-critical applications.

3.2 Timestamp Temporal Logic - TTL

As it is shown in the last sections, since any monitoring device has a level of measurement error, STL statements cannot be monitor correctly without considering such value in the statements. As a fact, this problem exists for any value of δ. Meaning with even a small value for δ, there is a level of inaccuracy. As it was shown, if we consider measurement error δ, then we can provide one-sided guarantee that the constraint is met. However, specifying the error of the measurement devices does not make sense in the application constraint. From the application perspective, it makes more sense to define an uncertainty value,ε, within which the constraints should be met. Engineers usually specify ε when it is the planned limit of acceptable *unintended* deviation from a nominal or theoretical dimension. It is from application perspective and related to the high-level operation of the system regardless of any measurement systems. If we make a relationship between measurement error and tolerance values of timing constraints, the connection to consider errors is made and then, the constraint can be correctly evaluated by a measurement system.

Timestamp Temporal Logic (TTL) provides a definitional extension of STL that more intuitively expresses the timing specifications of distributed CPS. TTL also allows for a more natural expression of timing tolerances. The language, provides a room for expressing timing constraints with considering tolerance values that makes the monitoring accurate.

Beside the TTL specifications, this chapter outlines a methodology to automatically generate logic code and programs to monitor the expressed timing constraints.

3.2.1 The Event Representation in TTL

Since there some operators in TTL that receive events on signals and evaluate the system timing requirements, it is needed to know to convert an analog signal to a sequence of events through threshold crossing.

Typically, an event is represented by a single point in the time domain. Accordingly, a signal event is constructed from a real-valued signal crossing a threshold value. A signal event, generated by threshold crossing on analog signals, is presented by a triplet, $\langle s, th, \nearrow \textit{ or } \searrow \rangle$, which is 1 ($\top$) at the time when the signal, s, crosses a threshold, th (crossing from below $\nearrow$ or from above $\searrow$), and 0 ($\bot$) everywhere else. A signal event can be singleton or repetitive. In a singleton signal event, there is only one event (ϕ) which is represented by a single timestamp while repetitive signal events are expressed by a sequence of events $\{\phi_1, \phi_2, ..., \phi_n\}_{(n \in \mathbb{N})}$ and can be represented by multiple timestamps.

Figure 3.4 shows the process to detect event on analog signals by threshold crossing. It depicts comparing two signals (s_1 and s_2) with their corresponding thresholds (th_1 and th_2) results in Boolean signals (ψ_1 and ψ_2) and applying differentiator operator ($\bowtie$) on Boolean signals results in instantaneous events (ϕ_1 and ϕ_2). The events on ψ_1 are repetitive hence, their sequence is shown by superscripts.

A Boolean signal can be divided into time intervals during which the value of the signal is true or false, indicated by $\mathcal{I}^+$ and $\mathcal{I}^-$ (figure 3.4 signals a and b). The occurrence of an event corresponding to a rising/falling edge is defined as the starting

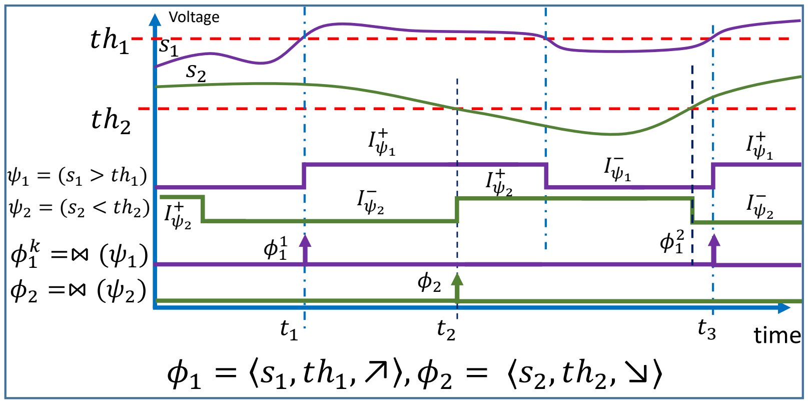


Figure 3.4: The Process to Events from Analog Signals by Threshold Crossing.

point of each positive interval ($I_i \in \mathcal{I}^+$). Each interval can be expressed by a set of rising and falling timestamps. Taking the rising edge can generate an event. As figure 3.4 illustrates, the rising edges on the Boolean signals, generated by threshold crossing, represent the events.

Definition 1. Differentiate operator, $\phi = \bowtie(\psi)$ converts a Boolean signal $\psi \in \mathbb{B}$ to a signal event where the value of ϕ is 1 when $\psi(t^+) \oplus \psi(t) \wedge \neg\psi(t) = \top$, and 0 otherwise. $\oplus$ is the XOR operator and, t^+ refers to the right neighborhood of signal at time t in continuous domain.

Extracting a signal event from a real-valued signal over the continuous time-domain is done by comparing the values of the signal with a threshold, th, and then, passing the output through the *Differentiate Operator* ($\bowtie$) in discrete time-domain. As it is depicted in figure 3.4, signals s_1 and s_2 are firstly converted into Boolean signals after comparing with their corresponding thresholds (th_1,th_2), and then by applying the differentiate operator, $\bowtie$ signal events, ϕ_1 and ϕ_2, are generated.

3.2.2 TTL Syntax

The TTL syntax is defined based on STL with extensions to enable distributed CPS with respect to absolute time, improved clarity, and considering tolerance values without substantial loss of meaning. TTL operators are built based on high-level operators that specify timing requirements on both the value of a formula and the occurrence time events. The output of TTL operators are finally a Boolean.

Definition 3.2.1. The comparison operator $\bigtriangledown$, is a mapping function from a real-valued signal to a Boolean value, where $\bigtriangledown \in \{>, =, <\}$.

Definition 3.2.2. The satisfaction relation $(s, t) \vDash \psi$, indicating that signal s satisfies φ starting from position t.

As the STL grammar, the **level-based** [6] temporal operators used in TTL are *Globally*, *Eventually*, and *Until* and defined inductively according to section 3.1.

Definition 3.2.3. Given the sets χ of events, and the set $\mathbf{V}$ of atomic propositions, the set $TTL_\chi(V)$ of TTL formulas (event-based) is inductively defined using the following grammar:

$\psi := v|\neg\psi|\psi_1 \wedge \psi_2|\mathcal{L}(\phi_1, \phi_2, \varepsilon)\triangledown l|\mathcal{C}(\phi_1, \phi_2, ...\phi_n, \varepsilon)|\mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)|\mathcal{F}(\phi, \varepsilon)\triangledown f$
$|\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)\triangledown p$

where $v \in \mathbf{V}$, $\phi_1, \phi_2, ..., \phi_n \in \chi$, $\triangledown \in \{<, >, =\}$ and $\delta, l, f, \varepsilon, \varepsilon_f, \varepsilon_p \in \mathbb{R}^+ + \{0\}$.

3.2.3 TTL Semantics

Before starting to explain the details of the language, there are some parameters that should be defined.

- s is a continuous signal over continuous time t.
- ψ is a Boolean signal over continuous time. It can be *true* (positive pules) or *false* (negative pulse) for a duration of time.

[6]Level-based operators receive Boolean signal values at each single moment (time step) and evaluate the constraint. On the other side, event-based timing constraints are operators that receive events on signals and are evaluated based on the events.

- δ is the sampling time to convert a continuous signal to discrete. It can be taken as measurement error as well because it is the maximum error in the discretization process.

- ϕ is an event signal that is *true* for a short time like δ. In discrete time domain, it is *true* for just one time step.

- $T(\phi)$ is a function that receives the event ϕ and returns its actual time of occurrence in continuous domain. It is a real number.

- $\tau(\phi)$ receives the event ϕ and returns the event's timestamp which is an integer number.

- ε is the user-defined tolerable error.

- $\bowtie$ is differentiator operator and converts a Boolean signal into signal event ($\varphi = \bowtie(\psi)$).

- δ is sampling time for discretization and assumed $\delta < 1s$.

As it is depicted in figure 3.4 the triplet $\langle s_1, th, \nearrow \rangle$ is equivalent to $\bowtie (s_1 > th)$ and the triplet $\langle s_2, th, \searrow \rangle$ is equivalent to $\bowtie (s_2 < th)$.

Now, the meaning of each operator is explained as follow:

Maximum Latency: $\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l, 0 < \varepsilon < l$

This proposition specifies that i) the events ϕ_1 and ϕ_2 are singleton [7] , ii) ϕ_1 occurs before ϕ_2, and iii) the difference between the actual occurrence of two events, ϕ_1 and ϕ_2 should be less than l with tolerance of ε. If $T(\phi)$ represents the actual occurrence time of event ϕ in real numbers, we have $0 < T(\phi_2) - T(\phi_1)$ from condition ii and

[7]they occur once in a while

$T(\phi_2) - T(\phi_1) < l - \varepsilon$ from condition iii. It should be less than $l - \varepsilon$ because we need to guarantee that the latency between those two events is less than l.

Figure 3.5.a depicts the how the latency constraint is calculated in continuous time.

Now, the question is when can we guarantee that $T(\phi_2) - T(\phi_1) < l - \varepsilon$ is satisfied in a discrete system?

Most CPS implementations and measurement systems sample signals (with sampling time of δ) and, therefore, capture a timestamp as the occurrence of the event. The actual time of the event is inferred from the timestamp within an error δ. The measurement error has several sources such as quantization, sampling time, analog to digital converter (ADC) resolution [8] . If $\tau(\phi)$ represents the ***integer*** timestamp at which the event ϕ is captured, and both $T(\phi)$ and $\tau(\phi)$ are initiated to zero when the system starts to operate, then $\tau(\phi) = \left\lceil \frac{T(\phi)}{\delta} \right\rceil$ is the relation between $T(\phi)$ and $\tau(\phi)$.

Therefore, we know that

$$T(\phi_1) = (\delta\tau(\phi_1) - \delta, \delta\tau(\phi_1)]$$

and similarly

$$T(\phi_2) = (\delta\tau(\phi_2) - \delta, \delta\tau(\phi_2)]$$

Hence, their subtraction to find the latency between two events ϕ_1 and ϕ_2 will be bounded between $\delta(\tau(\phi_2) - \tau(\phi_1)) - \delta$ and $\delta(\tau(\phi_2) - \tau(\phi_1)) + \delta$ [9] . In fact:

$$\delta(\tau(\phi_2) - \tau(\phi_1)) - \delta < T(\phi_2) - T(\phi_1) < \delta(\tau(\phi_2) - \tau(\phi_1)) + \delta \tag{3.1}$$

In order to be conservative, it is enough if we have

[8] Their effects are studied in chapter 6.

[9] Since the subtraction is calculated, the maximum error is δ.

$$T(\phi_2) - T(\phi_1) < l - \varepsilon \tag{3.2}$$

to guarantee the time difference between events is certainly less than l. Therefore, based on equations 3.2 and 3.1:

$$\delta\tau(\phi_2) - \delta\tau(\phi_1) - \delta < l - \varepsilon \tag{3.3}$$

Since we know that $0 < \delta\tau(\phi_2) - \delta\tau(\phi_1)$, $0 < l - \varepsilon + \delta$.

Hence to guarantee 3.2, we should test:

$$\tau(\phi_2) - \tau(\phi_1) < \frac{l - \varepsilon}{\delta} + 1 \tag{3.4}$$

and the condition is (by considering equations 3.1 and 3.3):

$$\varepsilon - \delta < l \tag{3.5}$$

On the other hand, in inequality 3.3, if $\varepsilon < \delta$, since we are adding a positive number to l ($\delta\tau(\phi_2) - \delta\tau(\phi_1) < l - (\varepsilon - \delta)$), it does not guarantee 3.3. Therefore, we should have $\delta \leq \varepsilon$. This relation between ε and δ makes sense because is equation 3.4, the added value (δ) due to the discretization process is compensated by ε.

In the above equations, 3.4 is a property of **measurement** and $\delta \leq \varepsilon$ is the property of **measurement system**.

Minimum Latency: $l < \mathcal{L}(\phi_1, \phi_2, \varepsilon)$

This specification implies that the time difference between the actual occurrence of event ϕ_2 and event ϕ_1 should be greater than l, with a tolerance of ε. In other words, if $T(\phi)$ is the actual occurrence of the event ϕ, then the specification implies that $l \pm \varepsilon < T(\phi_2) - T(\phi_1)$. Since the tolerance concept determines that $l + \varepsilon$ should be satisfied to guarantee the minimum latency, the relationship between actual time of events can be defined as: $l + \varepsilon < T(\phi_2) - T(\phi_1)$.

By considering equation 3.1:

$$l + \varepsilon - \delta < \delta(\tau(\phi_2) - \tau(\phi_1))$$

If $\varepsilon < \delta$, we cannot guarantee that the time difference between two events is greater than l. Therefore, $\delta \leq \varepsilon$.

Similar to the maximum latency specification, there will be an error of δ between the actual time and the captured timestamp of events. Hence we have:

$$\frac{l + \varepsilon}{\delta} - 1 < \tau(\phi_2) - \tau(\phi_1), \delta \leq \varepsilon \tag{3.6}$$

Exact Latency: $\mathcal{L}(\phi_1, \phi_2, \varepsilon) = l$

Exact latency specifies that the difference between the occurrence of two events ϕ_1 and ϕ_2 should be equal to l with a tolerance of ε. This means $T(\phi_2) - T(\phi_1) = l \pm \varepsilon$ or $l - \varepsilon < T(\phi_2) - T(\phi_1) < l + \varepsilon$. A monitoring system can ensure this specification by checking if $\frac{l-\varepsilon}{\delta} + 1 < \tau(\phi_2) - \tau(\phi_1) < \frac{l+\varepsilon}{\delta} - 1$. Again, the measurement system is considered to be able to evaluate the specification correctly if $\delta < \varepsilon$.

Chronological: $\mathcal{C}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$

It specifies that the event ϕ_i occurs before event ϕ_{i+1} $(1 \leq i \leq n)$, with a tolerance of ε. This means that $\varepsilon < T(\phi_{i+1}) - T(\phi_i)$. A measurement system with accuracy of δ will be able to ensure the specification by monitoring $\frac{\varepsilon}{\delta} - 1 < \tau(\phi_{i+1}) - \tau(\phi_i)$ only if $\delta < \varepsilon$.

Simultaneity: $\mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$

Specifies that the events ϕ_1 to ϕ_n occur at the same time with a tolerance of ε. This means that the time difference between each pair of events is less than ε, or by the

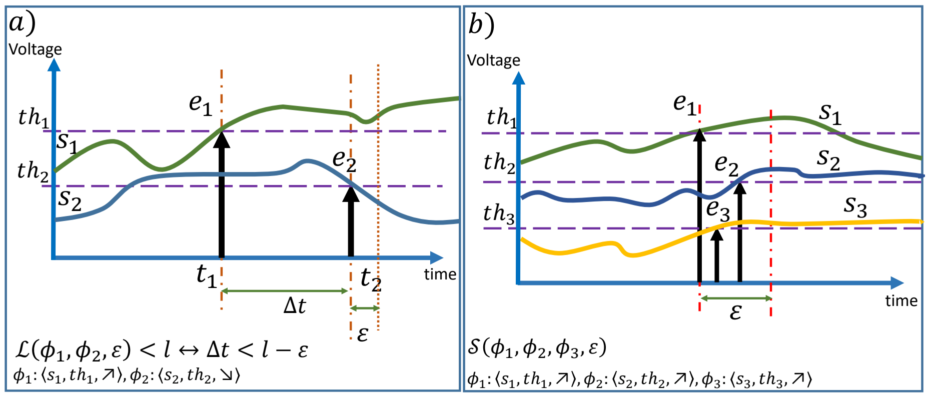


Figure 3.5: The Effect of Tolerance in *Latency* and *Simultaneity* Calculations.

other words

$$max\big(T(\phi_1), T(\phi_2), ..., T(\phi_n)\big) - min\big(T(\phi_1), T(\phi_2), ..., T(\phi_n)\big) < \varepsilon$$

A measurement system with accuracy of δ will be able to ensure the specification by monitoring

$$max\Big(\tau(\phi_1), \tau(\phi_2), ..., \tau(\phi_n)\Big) - min\Big(\tau(\phi_1), \tau(\phi_2), ..., ts(\phi_2)\Big) < \frac{\varepsilon}{\delta} + 1$$

The added δ value (everything is normalized by δ) is just to consider the measurement error value compensated by ε. Additionally, as before, the measurement is valid only if $\delta < \varepsilon$. Figure 3.5.b demonstrate the *Simultaneity* calculation.

Minimum Frequency: $f < \mathcal{F}(\phi, \varepsilon_f)$
specifies that the occurrence frequency of event ϕ [10] should be less than f with a tolerance of ε_f (in Hz) or, in time domain, $T(\phi^n) - T(\phi^{n-1}) < \frac{1}{f \pm \varepsilon_f}$ where ϕ^i corresponds to the ith occurrence of the event ϕ on the same signal. To simplify, we have: $T(\phi^n) - T(\phi^{n-1}) < \frac{1}{f + \varepsilon_f}$. Similar to latency, $\tau(\phi^n) - \tau(\phi^{n-1}) < \frac{1}{\delta(f+\varepsilon_f)} + 1$, $\delta < \frac{1}{\varepsilon_f}$.

Maximum Frequency: $\mathcal{F}(\phi, \varepsilon_f) < f$
specifies that the frequency of event ϕ should be less than f with a tolerance of ε_f

[10] ϕ is repetitive here and ϕ^i corresponds to the ith occurrence of the event ϕ.

(in Hz) or $\frac{1}{f \pm \varepsilon_f} < T(\phi^n) - T(\phi^{n-1})$. Simplified, this is $\frac{1}{f - \varepsilon_f} < T(\phi^n) - T(\phi^{n-1})$. Therefore, $\frac{1}{\delta(f-\varepsilon_f)} - 1 < \tau(\phi^n) - \tau(\phi^{n-1})$ where $\delta < \frac{1}{\varepsilon_f}$, $\varepsilon_f < f$.

Exact Frequency: $\mathcal{F}(\phi, \varepsilon_f) = f$

means that the occurrence frequency of the event ϕ should be equal to f with a tolerance of ε_f (in Hz) or $T(\phi^n) - T(\phi^{n-1}) = \frac{1}{f \pm \varepsilon_f}$ where ϕ^i corresponds to the i^{th} occurrence of event ϕ. Simplifying: $\frac{1}{f+\varepsilon_f} < T(\phi^n) - T(\phi^{n-1}) < \frac{1}{f-\varepsilon_f}$,$\varepsilon_f < f$.

Considering the measurement error of δ, the system must monitor $\frac{1}{\delta(f+\varepsilon_f)} + 1 < \tau(\phi^n) - \tau(\phi^{n-1}) < \frac{1}{\delta(f-\varepsilon_f)} - 1$, and the monitoring is valid only if $\delta < \frac{1}{\varepsilon_f}$,$\varepsilon_f < f$.

Minimum Phase: $p < \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)$

specifies that the phase difference between two repeating events ϕ_1, and ϕ_2 on two different event sources is greater than p, where, ε_f and ε_p are the tolerance in frequency and phase calculations. If events ϕ_1 and ϕ_2 occur at the **same frequency** (exact frequency) then *Phase* defines the desired latency between consequent events on two different event sources (ϕ_1 and ϕ_2). Hence, we must satisfy two conditions: i) $\mathcal{F}(\phi_1, \varepsilon_f) = \mathcal{F}(\phi_2, \varepsilon_f)$, and ii) $p - \varepsilon_p < T(\phi_2^n) - T(\phi_1^n)$.

From condition i)

$$|(T(\phi_1^n) - T(\phi_1^{n-1})) - (T(\phi_2^n) - T(\phi_2^{n-1}))| < \frac{1}{\varepsilon_f}$$

if we assume

$A : T(\phi_1^n) - T(\phi_1^{n-1})$ and $B : T(\phi_2^n) - T(\phi_2^{n-1})$ we have:

$-\frac{1}{\varepsilon_f} < A - B < \frac{1}{\varepsilon_f}$. Hence, there are two cases:

$$A < \frac{1}{\varepsilon_f} + B \tag{3.7}$$

and

$$B - \frac{1}{\varepsilon_f} < A \tag{3.8}$$

From 3.1, we know that

$$\delta(\tau(\phi_1^n) - \tau(\phi_1^{n-1})) - \delta < A < \delta(\tau(\phi_1^n) - \tau(\phi_1^{n-1})) + \delta$$

$$\delta(\tau(\phi_2^n) - \tau(\phi_2^{n-1})) - \delta < B < \delta(\tau(\phi_2^n) - \tau(\phi_2^{n-1})) + \delta$$

To be conservative, in 3.7, A should be in its maximum value and B should be in its Minimum value. Therefore:

$$\delta(\tau(\phi_1^n) - \tau(\phi_1^{n-1})) + \delta < \frac{1}{\varepsilon_f} + \delta(\tau(\phi_2^n) - \tau(\phi_2^{n-1})) - \delta$$

and in 3.8:

$$\delta(\tau(\phi_2^n) - \tau(\phi_2^{n-1})) + \delta < \frac{1}{\varepsilon_f} + \delta(\tau(\phi_1^n) - \tau(\phi_1^{n-1})) - \delta$$

Thus, the pre-conditions for satisfaction of *Phase* constraint is:

$$\tau(\phi_1^n) - \tau(\phi_1^{n-1}) - (\tau(\phi_2^n) - \tau(\phi_2^{n-1})) < \frac{1}{\delta\varepsilon_f} - 2 \tag{3.9}$$

and

$$\tau(\phi_2^n) - \tau(\phi_2^{n-1}) - (\tau(\phi_1^n) - \tau(\phi_1^{n-1})) < \frac{1}{\delta\varepsilon_f} - 2 \tag{3.10}$$

From condition ii), the specification implies that $\frac{p+\varepsilon_p}{\delta} - 1 < \tau(\phi_2^n) - \tau(\phi_1^n)$. Since the monitoring system must ensure that $p+\varepsilon_p < T(\phi_2^n) - T(\phi_1^n)$ is monitored correctly, the monitoring system will be implemented as $p' + \varepsilon_p' - 1 < \tau(\phi_2^n) - \tau(\phi_1^n)$ where $p' = \frac{p}{\delta}$ and $\varepsilon_p' = \frac{\varepsilon_p}{\delta}$. To check if the measurement system can evaluate the timing specification, $\delta < \frac{1}{\varepsilon_f}$ and $\delta < \varepsilon_p$ statements should hold.

Maximum Phase: $\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$

defines that the difference between timestamps of two events on two different event sources should be less than a certain value (p). For this specification, also the two frequencies should be the same. $T(\phi_2^n) - T(\phi_1^n) < p \pm \varepsilon$ where ϕ_1^i and ϕ_2^i correspond to the i^{th} occurrence of the events ϕ_1 and ϕ_2 respectively. Similar to the minimum phase, the monitoring system should monitor $\tau(\phi_2^n) - \tau(\phi_1^n) < p' - \varepsilon'_p + 1$.

Exact Phase: $\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p$

indicates that the difference between the timestamps of two events, ϕ_1^n and ϕ_2^n should be equal to p with a tolerance of ε_p where the frequency of occurrence of ϕ_1 and ϕ_2 must be equal (condition 1). This means $T(\phi_2^n) - T(\phi_1^n) = p \pm \varepsilon_p$ or $p - \varepsilon_p < T(\phi_2^n) - T(\phi_1^n) < p + \varepsilon_p$. Considering the error in the measurement system, we have $p' - \varepsilon'_p + 1 < \tau(\phi_2) - \tau(\phi_1) < p' + \varepsilon'_p - 1$.

Note that the events in *Frequency* and *Phase* formulas are necessarily periodic, whereas in the other timing specifications they should be singleton.

All aforementioned operators are semantically formulated in table 3.1 and in details in chapter 4.

Table 3.1: Satisfaction Relations and Language Semantics

1	$(\sigma, t) \models v$	iff $(\sigma)[t] = \top$
2	$(\sigma, t) \models \neg\psi$	iff $(\sigma, t) \not\models \psi$
3	$(\sigma, t) \models \psi_1 \wedge \psi_2$	iff $(\sigma, t) \models \psi_1$ and $(\sigma, t) \models \psi_2$
4	$(\sigma, t) \models l < \mathcal{L}_{[a,b]}(\phi_1, \phi_2, \varepsilon)$	iff $\exists t' \in [t+a, t+b]$ s.t. $(\sigma, t') \models \phi_1$ and $\exists t'' > t'$ s.t. $(\sigma, t'') \models \phi_2$ and $\frac{l+\varepsilon}{\delta} - 1 < (t'' - t')$.
5	$(\sigma, t) \models \mathcal{L}_{[a,b]}(\phi_1, \phi_2, \varepsilon) < l$	iff $\exists t' \in [t+a, t+b]$ s.t. $(\sigma, t') \models \phi_1$ and $\exists t'' > t'$ s.t. $(\sigma, t'') \models \phi_2$ and $(t'' - t') < \frac{l-\varepsilon}{\delta} + 1$.
6	$(\sigma, t) \models \mathcal{L}_{[a,b]}(\phi_1, \phi_2, \varepsilon) = l$	iff $\exists t' \in [t+a, t+b]$ s.t. $(\sigma, t') \models \phi_1$ and $\exists t'' > t'$ s.t. $(\sigma, t'') \models \phi_2$ and $\frac{l-\varepsilon}{\delta} + 1 \leq (t'' - t') \leq \frac{l+\varepsilon}{\delta} - 1$.
7	$(\sigma, t) \models \mathcal{C}_{[a,b]}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$	iff $\forall t_i \in [t + a, t + b], i = \{1, ..., n - 1\}$ s.t. $(\sigma, t_i) \models \phi_i$ and $(\sigma, t_{i+1}) \models \phi_{i+1}$ and $t_i < t_{i+1}$ and $\frac{\varepsilon}{\delta} - 1 < t_{i+1} - t_i$
8	$(\sigma, t) \models \mathcal{S}_{[a,b]}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$	iff $\forall t_i \in [t+a, t+b], i = \{1, ..., n\}$ s.t. $(\sigma, t_i) \models \phi_i$ and $\max\{t_i\} - \min\{t_i\} < \frac{\varepsilon}{\delta} + 1$
9	$(\sigma, t) \models f < \mathcal{F}_{[a,b]}(\phi, \varepsilon_f)$	iff $\forall t' \in [t+a, t+b], \exists t'' \in [t+a, t+b], t' < t''$ s.t. $(\sigma, t') \models \phi$ and $(\sigma, t'') \models \phi$ and $\nexists t''', t' < t''' < t''$ s.t. $(\sigma, t''') \models \phi$ and $(t'' - t') < \frac{1}{\delta(f+\varepsilon_f)} + 1$
10	$(\sigma, t) \models \mathcal{F}_{[a,b]}(\phi, \varepsilon_f) < f$	iff $\forall t' \in [t+a, t+b], \exists t'' \in [t+a, t+b], t' < t''$ s.t. $(\sigma, t') \models \phi$ and $(\sigma, t'') \models \phi$ and $\nexists t''', t' < t''' < t''$ s.t. $(\sigma, t''') \models \phi$ and $\frac{1}{\delta(f-\varepsilon_f)} - 1 < (t'' - t')$

11	$(\sigma, t) \models \mathcal{F}_{[a,b]}(\phi, \varepsilon_f) = f$	iff $\forall t' \in [t+a, t+b], \exists t'' \in [t+a, t+b], t' < t''$ s.t. $(\sigma, t') \models \phi$ and $(\sigma, t'') \models \phi$ and $\nexists t''', t' < t''' < t''$ s.t. $(\sigma, t''') \models \phi$ and $\frac{1}{\delta(f+\varepsilon_f)} + 1 \leq (t'' - t') \leq \frac{1}{\delta(f-\varepsilon_f)} - 1$
12	$(\sigma, t) \models p < \mathcal{P}_{[a,b]}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)$	iff $\forall t'_1 \in [t+a, t+b]$s.t. $(\sigma, t'_1) \models \phi_1$ and $\exists t''_1 \in [t+a, t+b]$ s.t. $t'_1 < t''_1$ and $(\sigma, t''_1) \models \phi_1$ and $\nexists t'''_1$s.t. $(\sigma, t'''_1) \models \phi_1$ and $\forall t'_2 \in [t+a, t+b]$s.t. $(\sigma, t'_2) \models \phi_2$ and $\exists t''_2 \in [t+a, t+b]$ s.t. $t'_2 < t''_2$ and $(\sigma, t''_2) \models \phi_2$ and $\nexists t'''_2$ s.t. $(\sigma, t'''_2) \models \phi_2$ and $t''_1 - t'_1 - (t''_2 - t'_2) < \frac{1}{\delta \varepsilon_f} - 2$ and $t''_2 - t'_2 - (t''_1 - t'_1) < \frac{1}{\delta \varepsilon_f} - 2$ and $(\frac{p}{\delta} + \frac{\varepsilon_p}{\delta} - 1) < t'_2 - t'_1$
13	$(\sigma, t) \models \mathcal{P}_{[a,b]}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$	iff $\forall t'_1 \in [t+a, t+b]$s.t. $(\sigma, t'_1) \models \phi_1$ and $\exists t''_1 \in [t+a, t+b]$ s.t. $t'_1 < t''_1$ and $(\sigma, t''_1) \models \phi_1$ and $\nexists t'''_1$ s.t. $(\sigma, t'''_1) \models \phi_1$ and $\forall t'_2 \in [t+a, t+b]$ s.t. $(\sigma, t'_2) \models \phi_2$ and $\exists t''_2 \in [t+a, t+b]$ s.t. $t'_2 < t''_2$ and $(\sigma, t''_2) \models \phi_2$ and $\nexists t'''_2$s.t. $(\sigma, t'''_2) \models \phi_2$ and $t''_1 - t'_1 - (t''_2 - t'_2) < \frac{1}{\delta \varepsilon_f} - 2$ and $t''_2 - t'_2 - (t''_1 - t'_1) < \frac{1}{\delta \varepsilon_f} - 2$ and $t'_2 - t'_1 < (\frac{p}{\delta} - \frac{\varepsilon_p}{\delta} + 1)$
14	$(\sigma, t) \models \mathcal{P}_{[a,b]}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p$	iff $\forall t'_1 \in [t+a, t+b]$s.t. $(\sigma, t'_1) \models \phi_1$ and $\exists t''_1 \in [t+a, t+b]$ s.t. $t'_1 < t''_1$ and $(\sigma, t''_1) \models \phi_1$ and $\nexists t'''_1$ s.t. $(\sigma, t'''_1) \models \phi_1$ and $\forall t'_2 \in [t+a, t+b]$ s.t. $(\sigma, t'_2) \models \phi_2$ and $\exists t''_2 \in [t+a, t+b]$ s.t. $t'_2 < t''_2$ and $(\sigma, t''_2) \models \phi_2$ and $\nexists t'''_2$s.t. $(\sigma, t'''_2) \models \phi_2$ and $t''_1 - t'_1 - (t''_2 - t'_2) < \frac{1}{\delta \varepsilon_f} - 2$ and $t''_2 - t'_2 - (t''_1 - t'_1) < \frac{1}{\delta \varepsilon_f} - 2$ and $(\frac{p}{\delta} - \frac{\varepsilon_p}{\delta} + 1) \leq t'_2 - t'_1 \leq (\frac{p}{\delta} + \frac{\varepsilon_p}{\delta} - 1)$

Chapter 4

TTL REASONING SYSTEM

As it is discussed earlier, one practical approach to verify the temporal behavior of Cyber-physical Systems is run-time monitoring. While monitoring based assurance schemes make sense, there are still several practical challenges in enabling them:

i) Specified requirements may not be consistent: In the effort to address safety concerns, designers often write down a lot of safety specifications. One important task is to determine if all the specifications are consistent.
ii) Complexity of monitoring logic: Since CPS can be large and complex, and designers often over-specify the requirements, the monitoring logic can be very large, which in turn requires tremendous resources to be able to perform online monitoring. Limitation of resources is the main reason offline monitoring approaches are more broadly used.
iii) Handling timing requirements specified over inaccessible signal: Many CPS designers define specifications involving some events or signals that are not directly accessible through measurement.

Without a major change in the design which may affect the system timing, such specifications are impossible to monitor.

One way to address above challenges is to develop a deductive reasoning framework for timing specifications. A deductive reasoning framework is a set of rules and transformations that allows for establishing a meaningful relationship between individual statements [Hoare, 1969]. Researchers have proposed deductive reasoning systems for LTL [Bolotov et al., 2006], Computation Tree Logic (CTL) [Pnueli and Kesten, 2002], and Propositional Projection Temporal Logic (PPTL) [Duan and Zhang, 2008]. However, these logics are not sufficient to express the timing specifications of CPS, since they are defined over Boolean values and discrete time – while CPS operate on real signal values and over continuous time. The objective of this chapter is to augment the temporal specifications with a deductive reasoning framework for TTL.

To demonstrate the usefulness of the approach two case studies have been implemented i) a flying paster application – and ii) an intersection management system for connected autonomous vehicles, which schedules the safe and efficient passage of autonomous vehicles through a signal-free intersection. The safety and performance-related timing specifications are specified in TTL for both applications first, and then the proposed reasoning is applied on various scenarios.

4.1 Background

Before having the details for the reasoning framework, it is needed to have some sort of definitions for SUT and monitoring framework.

4.1.1 Continuous-Time Signals

A continuous signal s maps the continuous time domain (dense time) to a real-valued domain.

Definition 4.1.1. Signal s is a map as $s : \mathbb{T} \rightarrow \mathbb{R}$ where $\mathbb{T}$ is the set of non-negative real numbers as time, $\mathbb{R}_{\geq 0}$, and $\mathbb{R}$ is the value of analog signals.

In monitoring CPS, since it is usually based upon finite traces [Maler and Nickovic, 2004] and we are interested in Boolean values, s is elaborated as $s : \mathbb{R}_{\geq 0} \rightarrow \mathbb{B}$ [1] . The

[1] $\mathbb{B}$ is Boolean values containing *true* and *false*

interval for the entire signal is $I = [0, r)$, and its value at time t where $t \in \mathbb{R}_{\geq 0}$ is shown by $s(t)$. The signal length is r expressed by $|s| = r$.

4.1.2 Discrete-Time Signals

A discrete-time signal σ is a sequence of samples of a continuous-time signal s. The discrete signal is σ and $\sigma : \mathbb{N} \rightarrow \mathbb{B}$. The value of σ at position i by $\sigma[i]$ where $i \in \mathbb{N}$. In order to extract the signal value at time t, we need just to multiply the sample number to the *"sampling time"*, δ, $(t = i \times \delta)$.

4.1.3 Events on Signals

In Temporal Logic, *Event* can be defined as useful and relevant patterns of changes[Allen and Ferguson, 1994]. Events in CPS are driven by the process, the system signals, the physical environment, etc. We can generally determine a pattern of ***changes*** over time for an event. TTL augments the event specification through explicit definition of the change thresholds in the relevant system parameters. For Boolean signals, changing the value from *true* $(\top)$ to *false* $(\bot)$ and from $\bot$ to $\top$ are detected as events and we call them *"rising"* and *"falling"* events, respectively.

To have a logic definition for event, we enumerate all conditions who cause the certain event in a single proposition. $\phi : action(c_1, ..., c_n)$ expresses the action that initiates an event, ϕ, where c_1 through c_n are conditions at the time the event is triggered.

Definition 4.1.2. $\phi@t$ denotes the occurrence of event ϕ at time "t" when all conditions $c_1, c_2, ..., c_n$ are satisfied. The function $T(\phi)$ returns i as its occurring time (real number).

Definition 4.1.3. In discrete time, $\phi@i$ is the event occurred at time step i, and function $\tau(\phi)$ returns i as its *timestamp*.

After having the required definitions for continuous and discrete signals, Boolean signals, events on signals, continuous and discrete time, and timestamps, everything is ready to explain the axioms and rules for TTL.

Using a deduction framework, a CPS designer can define timing specifications in TTL and then prove new statements to evaluate correctness of temporal behavior. TTL should contain arithmetic axioms (e.g. $a + b = a \leftrightarrow b = 0$) as well as logical axioms and rules (e.g. $p \rightarrow q \leftrightarrow \neg p \vee (p \wedge q)$) as the primitives in the deduction process.

4.2 TTL Arithmetic Axioms

TTL is defined over the natural numbers, typically used to express time (or timestamps), constants, variables, and rational numbers to represent the values in the continuous world. Based on those two categories, a set of arithmetic axioms are defined. Table 4.1 represents some axioms utilized in this chapter. Note that $x, y, z, w, a, b, c, d \in \mathbb{N}$

Definition 4.2.1. Functions: $g(x_1, ..., x_n)$, each argument x_i is an input and g (in lower case) is a function with n variables as arguments. Functions represent a subset of relations where there is only one "*value*" for any set of "*inputs*". If $n = 0$, g is a *constant*.

4.3 TTL Axioms and Logic Rules

In this section, the axioms and logic rules are introduced for TTL using operators defined in chapter 3 and other logic operators (such as conjunction, disjunction, implication, etc.) from [Prawitz, 2006]. As it is shown earlier, timing specifications,

Table 4.1: TTL Arithmetic Axioms.

Rule Name	Arithmetic Axiom
A1	$x + 0 \quad = \quad x$
A2	$(x + y) + z \quad = \quad x + (y + z)$
A3	$x + (-x) \quad = \quad 0$
A4	$x < y, z < w \quad \rightarrow \quad x + z < y + w$
A5	$x < y \quad \leftrightarrow \quad -y < -x$
A6	$x < z - y \quad \leftrightarrow \quad x < z + y$
A7	$x < y, z = x \quad \leftrightarrow \quad z < y$
A8	$x < y \quad \rightarrow \quad x - z < y$
A9	$x < y + z \quad \leftrightarrow \quad x - y < z$
A10	$x \leq y \quad \leftrightarrow \quad x < y + 1$
Q1	$\frac{a}{b} = \frac{c}{d} \quad \leftrightarrow \quad \frac{d}{b} = \frac{c}{a}; a, b, c, d \neq 0$
Q2	$\frac{a}{b} < \frac{c}{d} \quad \leftrightarrow \quad \frac{d}{c} < \frac{b}{a}; a, b, c, d \neq 0$
Q3	$\frac{a}{b} < \frac{c}{d} \quad \leftrightarrow \quad \frac{a.d}{b} < c; b, d \neq 0$
Q4	$\frac{a}{b} < \frac{c}{d} \quad \leftrightarrow \quad a < \frac{b.c}{a}; b, d \neq 0$

in TTL, express the relations between the occurrences of ***events***. $\mathcal{L}, \mathcal{S}, \mathcal{C}, \mathcal{F}, \mathcal{P}$ are the event-based timing operators in TTL that receive events and then evaluate the desired timing specifications utilizing their timestamps. Hence, it is required to know how to logically extract an event's timestamp.

Logic definition for Timestamp

Timing specifications are calculated using event timestamps. In the TTL formalism, the timestamp follows the "@" symbol in an event (section 4.1.3). By natural deduction, it is possible to extract timestamp of an event using a function. Indeed, $\phi@\tau$

shows *event* ϕ at time-step T and function "$\tau(\phi@i)$" receives the event ϕ and returns its happening timestamp, i.

Definition 4.3.1. *Introduction* and *Elimination*: Rules in a logic come in one of two flavors, i) *Introduction* or ii) *Elimination* rules. Introduction rules introduce the use of a logical operator, and elimination rules eliminate it.

The following example is an instance for introduction and Elimination rules.

The rules for "*conjunction*" ($\wedge$) are: $\frac{\phi,\ \psi}{\phi\wedge\psi} \wedge I$, $\frac{\phi\wedge\psi}{\psi} \wedge LE$, $\frac{\phi\wedge\psi}{\phi} \wedge RE$ where $\wedge I$ rule introduces the conjunction on two propositions ϕ and ψ, and $\wedge LE$ and $\wedge RE$ rules show the elimination of the left and right operands respectively. In the deduction below, the first rule introduces the logic "*conjunction*" between ϕ and ψ – called $\wedge I$. Rule $\wedge LE$ eliminates the left operand and deduce if "$\phi \wedge \psi$ *then* ψ ". Each row in a proof comprises a sequence number, rule of inference, and the prior line or lines of the proof that license that rule. For instance to prove: "$P \wedge Q, Q \wedge R \Rightarrow P \wedge R$" we have:

$$\begin{array}{r|ll}
1 & P \wedge Q & \\
2 & Q \wedge R & \\
\cline{2-2}
3 & P & \wedge\text{RE, 1} \\
4 & R & \wedge\text{LE, 2} \\
5 & P \wedge R & \wedge\text{I, 3, 4}
\end{array}$$

4.3.1 TTL Axioms:

Now, the required definitions to be used in natural deduction proofs are available. Table 4.2 demonstrates the TTL axioms. The logic behind the table is the TTL semantics but they are represented as the introduction and elimination rules. By those axioms, we can have proofs in natural deduction style. Keep in mind that, for all existing formulas, $\varepsilon < l$, $\delta < \varepsilon$, $\delta < \frac{1}{\varepsilon_f} < \varepsilon_p$, $\varepsilon_f < f$, and $\varepsilon_p < p$.

In order to understand the rules listed in Table 4.2, there is a proof for *Minimum Latency* rule in Theorem 4.3.2 in section 4.4.

Table 4.2: The Introduction and Elimination Rules for TTL Timing Specifications.

	TTL Specification	**Introduction Rule**	**Elimination Rule**
1	$l < \mathcal{L}(\phi_1, \phi_2, \varepsilon)$	$\dfrac{0<\tau(\phi_2)-\tau(\phi_1)\wedge(\frac{l+\varepsilon}{\delta}-1)<\tau(\phi_2)-\tau(\phi_1)}{l<\mathcal{L}(\phi_1,\phi_2,\varepsilon)}\text{LMI}$	$\dfrac{l<\mathcal{L}(\phi_1,\phi_2,\varepsilon)}{0<\tau(\phi_2)-\tau(\phi_1)\wedge(\frac{l+\varepsilon}{\delta}-1)<\tau(\phi_2)-\tau(\phi_1)}\text{LME}$
2	$\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l$	$\dfrac{0<\tau(\phi_2)-\tau(\phi_1)\wedge\tau(\phi_2)-\tau(\phi_1)<\frac{l-\varepsilon}{\delta}+1}{\mathcal{L}(\phi_1,\phi_2,\varepsilon)<l}\ \text{LXI}$	$\dfrac{\mathcal{L}(\phi_1,\phi_2,\varepsilon)<l}{0<\tau(\phi_2)-\tau(\phi_1)\wedge\tau(\phi_2)-\tau(\phi_1)<\frac{l-\varepsilon}{\delta}+1}\text{LXE}$
3	$\mathcal{L}(\phi_1, \phi_2, \varepsilon) = l$	$\dfrac{\frac{l-\varepsilon}{\delta}+3\leq\tau(\phi_2)-\tau(\phi_1)\wedge\tau(\phi_2)-\tau(\phi_1)\leq\frac{l+\varepsilon}{\delta}-3}{\mathcal{L}(\phi_1,\phi_2,\varepsilon)=l}\text{LEI}$	$\dfrac{\mathcal{L}(\phi_1,\phi_2,\varepsilon)=l}{\frac{l-\varepsilon}{\delta}+1\leq\tau(\phi_2)-\tau(\phi_1)\wedge\tau(\phi_2)-\tau(\phi_1)\leq\frac{l+\varepsilon}{\delta}-1}\text{LEE}$
4	$\mathcal{C}(\phi_1, ..., \phi_n, \varepsilon)$	$\dfrac{\bigwedge\limits_{k=1}^{n-1}(\frac{\varepsilon}{\delta}-1)<\tau(\phi_{k+1})-\tau(\phi_k)}{\mathcal{C}(\phi_1,...,\phi_n,\varepsilon)}\text{CHI}$	$\dfrac{\mathcal{C}(\phi_1,...,\phi_n,\varepsilon)}{\bigwedge\limits_{k=1}^{n-1}(\frac{\varepsilon}{\delta}-1)<\tau(\phi_{k+1})-\tau(\phi_k)}\text{CHE}$
5	$\mathcal{S}(\phi_1, \phi_2..., \phi_n, \varepsilon)$	$\dfrac{\max\limits_{\tau(\phi_k),1\leq k\leq n}-\min\limits_{\tau(\phi_k),1\leq k\leq n}<\frac{\varepsilon}{\delta}+1}{\mathcal{S}(\phi_1,...,\phi_n,\varepsilon)}\text{SMI}$	$\dfrac{\mathcal{S}(\phi_1,...,\phi_n,\varepsilon)}{\max\limits_{\tau(\phi_k),1\leq k\leq n}-\min\limits_{\tau(\phi_k),1\leq k\leq n}<\frac{\varepsilon}{\delta}+1}\text{SME}$
6	$f < \mathcal{F}(\phi, \varepsilon_f)$	$\dfrac{\tau(\phi^i)-\tau(\phi^{i-1})<\frac{1}{\delta(f+\varepsilon_f)}+1}{f<\mathcal{F}\big(\phi,\varepsilon_f\big)}\text{FMI}$	$\dfrac{f<\mathcal{F}\big(\phi,\varepsilon_f\big)}{\tau(\phi^i)-\tau(\phi^{i-1})<\frac{1}{\delta(f+\varepsilon_f)}+1}\text{FME}$
7	$\mathcal{F}(\phi, \varepsilon_f) < f$	$\dfrac{\frac{1}{\delta(f-\varepsilon_f)}-1<\tau(\phi_i)-\tau(\phi_{i-1})}{\mathcal{F}\big(\phi,\varepsilon_f\big)<f}\text{FXI}$	$\dfrac{\mathcal{F}(\phi,\varepsilon_f)<f}{\frac{1}{\delta(f-\varepsilon_f)}-1<\tau(\phi_i)-\tau(\phi_{i-1})}\text{FXE}$
8	$p < \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)$	$\dfrac{\begin{array}{c}\dfrac{}{\mathcal{F}\big(\phi_1,\varepsilon_f\big)=\mathcal{F}\big(\phi_2,\varepsilon_f\big)}u \\ \vdots \\ \frac{p+\varepsilon_p}{\delta}-1<\tau(\phi_2^i)-\tau(\phi_1^i)\end{array}}{\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)<p}\text{PMI}$	$\dfrac{p<\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)}{\forall i\in\mathbb{N},\mathcal{F}\big(\phi_1,\varepsilon_f\big)=\mathcal{F}\big(\phi_2,\varepsilon_f\big)\wedge(\frac{p+\varepsilon_p}{\delta}-1)<\tau(\phi_2^i)-\tau(\phi_1^i)}\text{PME}$
9	$\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$	$\dfrac{\begin{array}{c}\dfrac{}{\mathcal{F}\big(\phi_1,\varepsilon_f\big)=\mathcal{F}\big(\phi_2,\varepsilon_f\big)}u \\ \vdots \\ \tau(\phi_2^i)-\tau(\phi_1^i)<\frac{p-\varepsilon_p}{\delta}+1\end{array}}{\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)<p}\text{PXI}$	$\dfrac{\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)<p}{\forall i\in\mathbb{N},\mathcal{F}\big(\phi_1,\varepsilon_f\big)=\mathcal{F}\big(\phi_2,\varepsilon_f\big)\wedge\tau(\phi_2^i)-\tau(\phi_1^i)<\frac{p-\varepsilon_p}{\delta}+1}\text{PXE}$

4.3.2 TTL Rules

There are some TTL rules here that express temporal specifications in terms of latency.

- LE (Exact Latency): $\vdash \mathcal{L}(\phi_1, \phi_2, \varepsilon) = l \rightarrow\vdash (\neg(l < \mathcal{L}(\phi_1, \phi_2, \varepsilon))\wedge\neg(\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l))$.

  This rule is about *Exact Latency.*The conjunction of maximum and minimum latencies with the same l and ε values makes an exact latency.

- SL (Simultaneity to Latency): $\vdash \mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon) \rightarrow \vdash \mathcal{L}(\phi_i, \phi_j, \varepsilon_l) < \varepsilon, \quad where$ $\forall i, j \in \{1, 2, ..., n\}, \varepsilon_r < \varepsilon, i < j.$

  If *Simultaneity* between n events for ε is $\top$, the latency between any pair of events is less than ε.

- CL (Chronological to Latency): $\vdash \mathcal{C}(\phi_1, \phi_2, ...\phi_n, \varepsilon) \rightarrow \vdash \varepsilon < \mathcal{L}(\phi_i, \phi_{i+1}) \quad where$ $i \in \mathbb{N}, 1 \leq i \leq n - 1.$

  If *Chronological* between n events for ε is $\top$, the latency between any pair of adjacent events in the statement is more than ε.

- FL (Frequency to Latency): $\vdash \mathcal{F}(\phi, \varepsilon_f) < f \rightarrow \vdash \frac{1}{f+\varepsilon_f} < \mathcal{L}(\phi^{k-1}, \phi^k, \varepsilon_l) \quad where$ $\varepsilon_l < \frac{1}{\varepsilon_f}, k \in \mathbb{N}, k > 1.$

  If *Maximum Frequency* is $\top$, the latency between any pair of adjacent events is more than a certain value.

- PL (Phase to Latency): $\vdash \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p \rightarrow \vdash \Big(\mathcal{F}(\phi_1, \varepsilon_f) = \mathcal{F}(\phi_2, \varepsilon_f)\Big) \wedge$ $\Big(\mathcal{L}(\phi_1^k, \phi_2^k, \varepsilon_p) < p\Big), \forall k \in \mathbb{N}, 1 \leq k \leq n.$

  If *Maximum Phase* is $\top$ for two event signals, their frequencies are equal and the latency of events on two signals are less than a specific value.

- FE (Exact Frequency): $\vdash \mathcal{F}(\phi, \varepsilon_f) = f \rightarrow \vdash \neg\Big(f < \mathcal{F}(\phi, \varepsilon_f)\Big) \wedge \neg\Big(\mathcal{F}(\phi, \varepsilon_f) <$ $f\Big).$

  Exact Frequency is the conjunction of maximum and minimum *Frequency* operators.

- PE (Exact Phase): $\vdash \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p \rightarrow \vdash \neg\Big(p < \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)\Big) \wedge$ $\neg\Big(\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p\Big)$

  Exact Phase is the conjunction of maximum and minimum *Phase* operators.

In Theorem 4.3.1, there is a proof for LE rule and the rest of proofs including *Soundness* are transferred to the Appendix B.

Theorem 4.3.1. *The exact latency equals to the conjunction of Maximum and Minimum Latency constraints.*

$$\vdash \mathcal{L}(\phi_1, \phi_2, \varepsilon) = l \rightarrow \vdash (\neg(l < \mathcal{L}(\phi_1, \phi_2, \varepsilon)) \wedge \neg(\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l))$$

Proof. Consider $A : \mathcal{L}(\phi_1, \phi_2, \varepsilon) = l$, $B : \neg(l < \mathcal{L}(\phi_1, \phi_2, \varepsilon))$, and $\neg(C : \mathcal{L}(\phi_1, \phi_2, \varepsilon) < l)$.

For term A, if $\nvdash \mathcal{L}(\phi_1, \phi_2, \varepsilon) = l$, regardless of B and C, the entire formula is $\top$. Hence, we suppose A is $\top$ so we need to prove $B \wedge C = \top$.

For B, $0 < \tau(\phi_2) - \tau(\phi_1)$ and $\neg(\frac{l+\varepsilon}{\delta} - 1 < \tau(\phi_2) - \tau(\phi_1))$. For C, $0 < \tau(\phi_2) - \tau(\phi_1)$ and $\neg(\tau(\phi_2) - \tau(\phi_1) < \frac{l-\varepsilon}{\delta} + 1)$. Assume, $t' = \tau(\phi_1)$ and $t'' = \tau(\phi_2)$, $\frac{l-\varepsilon}{\delta} + 1 \leq t'' - t' \leq \frac{l+\varepsilon}{\delta} - 1$ which is the same as the semantic definition for exact latency in Table 3.1, line 6.

□

In Theorem 4.3.2, there is a proof for the existing Minimum Latency axiom (LMI) in Table 4.2. It shows how we can validate each rule.

Theorem 4.3.2. *The introduction rule for Minimum Latency is valid. i.e.,*
$\frac{(0<\tau(\phi_2)-\tau(\phi_1))\wedge(\frac{l+\varepsilon}{\delta}-1<\tau(\phi_2)-\tau(\phi_1))}{l<\mathcal{L}(\phi_1,\phi_2,\varepsilon)}$ *LMI is true*

Proof. Let ψ be a formula of $TTL_\chi(V)$, ψ is *universally* valid or a *tautology* if $(\sigma, t) \models \psi$ for every $t \in \mathbb{N}$. Therefore, regarding the syntax for *Minimum Latency*, we hae:

Suppose $(\sigma, t) \models \Big((0 < \tau(\phi_2) - \tau(\phi_1)) \wedge (\tau(\phi_2) - \tau(\phi_1) < \frac{l+\varepsilon}{\delta} - 1)\Big)$
iff $(\sigma, t) \models 0 < \tau(\psi_2) - \tau(\phi_1)$ and $(\sigma, t) \models \tau(\phi_2) - \tau(\phi_1) < \frac{l+\varepsilon}{\delta} - 1$ iff $0 < \tau(\phi_2) - \tau(\phi_1)$ and $\tau(\phi_2) - \tau(\phi_1) < l + \varepsilon$ considering $t' = \tau(\phi_1)$ and $t'' = \tau(\phi_2)$, since $0 < \tau(\phi_2) - \tau(\phi_1)$, we have $t' < t''$. Therefore, $\exists t' \in \mathbb{N}$ s.t. $(s, t') \models \phi_1$ and $\exists t'' > t'$ s.t. $(s, t'') \models \phi_2$

and $t'' - t' < \frac{l+\varepsilon}{\delta} - 1$ that based on the minimum latency semantics, implies $l < \mathcal{L}(\phi_1, \phi_2, \varepsilon)$ $\square$

4.4 Natural Deduction for TTL

The TTL deduction system is quite similar to applying *natural deduction* on propositional logic. A proof of proposition P in natural deduction starts from axioms and assumptions. Every step in the proof is an instance of an inference rule with expressions of the appropriate syntactic class.

Having the hypothetical judgments or *reasoning from assumptions* is required in the logic because they make the use of assumptions in proving process. This judgement contains two parts: i) the top sentence, called assumption, and ii) bottom sentence which is evaluated statement. One example of such statement in TTL is maximum Phase axiom. It is a *hypothetical judgments* using PXI axiom:

Phase Rules $\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$: Since this specification contains the frequency ***condition*** (see chapter 3), *hypothetical judgments*, it can be used as hypothesis in natural deduction. The Introduction (PXI) rule is defined as follows:

$$\frac{}{\mathcal{F}(\phi_1,\varepsilon_f)=\mathcal{F}(\phi_2,\varepsilon_f)}u$$

$$\vdots$$

$$\frac{\tau(\phi_2^i) - \tau(\phi_1^i) < \frac{p-\varepsilon_p}{\delta} + 1}{\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)<p}\text{PXI}$$

Elimination (PXE) rule for maximum phase specification is defined as

$$\frac{\mathcal{P}(\phi_1,\phi_2,\varepsilon_f,\varepsilon_p)<p}{\forall i, \mathcal{F}(\phi_1,\varepsilon_f)=\mathcal{F}(\phi_2,\varepsilon_f) \quad \wedge \quad \tau(\phi_2^i)-\tau(\phi_1^i)<\frac{p-\varepsilon_p}{\delta}+1}\text{PXE.}$$

The introduction rule states that *if* two events (ϕ_1, ϕ_2) occur periodically at the same frequency, the time difference between two consecutive events should be monitored to evaluate the given offset. The offset is compared with a value 'p' to create a predicate. The introduction (PXI) and elimination (PXE) rules for phase determine the maximum phase offset specification. The Rule for *Minimum* and *Exact* phase offset specifications can be similarly deduced (see Table 4.2 and Appendix B).

4.4.1 Natural Deduction Reasoning on TTL statements

In natural deduction, the reasoning process starts with premises, it applies rules of inference to derive conclusions and then strings such derivations together to form logical proofs. The idea is quite simple. Deducing a new statement from the premises, axioms, rules and/or existing previous derived statements by referring them using their name on the right and the line numbers on the left. Fitch style is chosen as the proof style since it is particularly popular in the Logic community, powerful as many other proof systems and is far simpler to use. Lemma 4.4.1 in this section demonstrates an example of reasoning process in Fitch style.

Lemma 4.4.1. Latency tolerance is cumulative.
$\mathcal{L}(\phi_1, \phi_2, \varepsilon_1) < l_1, \mathcal{L}(\phi_2, \phi_3, \varepsilon_2) < l_2 \implies \mathcal{L}(\phi_1, \phi_3, \varepsilon_1 + \varepsilon_2) < l_1 + l_2 + \delta$

Proof. The name of each rule is shown on the right side of steps and the deduced statement is written on the left side in each line:

1	$\mathcal{L}(\phi_1, \phi_2, \varepsilon_1) < l_1$	
2	$\mathcal{L}(\phi_2, \phi_3, \varepsilon_2) < l_2$	
3	$\tau(\phi_2) - \tau(\phi_1) < \frac{l_1 - \varepsilon_1}{\delta} + 1$	LXE, 1
4	$\tau(\phi_3) - \tau(\phi_2) < \frac{l_2 - \varepsilon_2}{\delta} + 1$	LXE, 2
5	$\tau(\phi_3) + \tau(\phi_2) - \tau(\phi_2) - \tau(\phi_1) < \frac{l_1 - \varepsilon_1 + l_2 - \varepsilon_2}{\delta} + 2$	A4, 3, 4
6	$\tau(\phi_3) - \tau(\phi_1) < \frac{l_1 - \varepsilon_1 + l_2 - \varepsilon_2}{\delta} + 2$	A3, 5
7	$\tau(\phi_3) - \tau(\phi_1) < \frac{(l_1 + l_2) - (\varepsilon_1 + \varepsilon_2)}{\delta} + 2$	A2, 6
8	$\mathcal{L}(\phi_1, \phi_3, \varepsilon_1 + \varepsilon_2) < l_1 + l_2 + \delta$	LXI, 7

$\square$

The first two premises are the latency between ϕ_1 and ϕ_2, and ϕ_2 and ϕ_3. Lines 3 and 4 eliminate the *Latency* specification by LXE in table 4.2. In lines 5-7, the proof uses the axioms in table 4.1. Similarly, line 8 is deduced from line 7 and LXI in table 4.2.

4.4.2 The TTL Reasoning Features

The reasoning system has the capability to provide four distinct capabilities. As it is showed in Lemma 4.4.1, by applying introduction and elimination rules on two *Latency* predicates a new TTL statement can be deduced. Since we can implement just one statement and deduce about the correctness of two other predicates, this approach can reduce the size of monitoring circuit and, thus, a more efficient run-time verification approach. As a summary, the following features are the outcome of the reasoning system.

Consistency Checking of Timing Specifications

The approach for consistency checking is very simple and straightforward. For monitoring a temporal behavior, there is a set of TTL hypothesises to be verified and for each, there is at least an Elimination rule (as table 4.2 illustrates). Therefore, it is possible to convert all hypothesis to their corresponding mathematical (inequality) formulas. Then, by applying the axioms in table 4.1, new formulas can be deduced and then converted to TTL statements using Introduction rules. Finally, if the produced statements contradict any of hypothesis, it results that there is an inconsistency and the deduced statements cannot satisfy all hypothesises.

As a practical example, there is a proof showing an inconsistency in the Case Study in chapter 7, section 7.4.

Considering Error Sources for Tolerance Specification

In all axioms and natural deduction rules in section 4.3, the user-defined acceptable tolerance and the measurement error are considered. Taking such variables into account make the monitoring system more accurate, predictable and practical in implementation. The example in Lemma 4.4.1 shows the way to accumulate the uncertainties in the proof steps and their impact on the final statements. In chapter 7, section 7.4.3, the case study for flying paster demonstrates such benefits.

Simplifying the Monitoring Logic

One of the most important features in the proposed method is its ability to simplify a set of temporal specifications to reach a higher degree of efficiency in the monitoring circuit. Timestamp-based Monitoring Approach (TMA [2]), proposes the efficient al-

[2]Explained in chapter 5

gorithms for single operators, however, it needs a way to reduce the required space on FPGA boards for combination of several temporal constraints. The reasoning framework provides such ability since by simple steps we can combine related statements.

After applying the elimination rules on TTL statements, because they comprise simple mathematical formulas, there is an ability to find the common variables in different statements. Then, by utilizing table 4.1 and considering the common parameters, it is possible to have a combination of several statements and finally reduce the number of operators. As a bright example, having two *Latency* operators (hypothesises) in Lemma 4.4.1, the deduction rules, and axioms can reduce the number of statements to only one *Latency*. Intuitively, implementing one operator needs less area and power on FPGA than two. In chapter 7, section 7.4.4 contains two case studies in different applications (flying paster and connected autonomous vehicles) demonstrating the capability to simplify a set of statements.

Reasoning about Unobservable Events

TTL reasoning system has the ability to give some understanding about unobservable signals. Applying the rules on hypothesises can extract the signals from the operators and then make new statements. In section 7.4.5, there is an example to illustrate this capability.

Chapter 5

TIMESTAMP-BASED MONITORING APPROACH TO MONITOR TIMING SPECIFICATIONS OF CPS

This chapter proposes an efficient online approach for monitoring the timing constraints of CPS that is called **T**imestamp-based **M**onitoring **A**pproach (**TMA**). The key improvement is rather than evaluating a constraint at each sampling period, TMA only computes the constraint satisfaction at the occurrence of relevant events extracted from monitored signals. The approach just uses the timestamps of events on signals and calculates the satisfaction of timing specifications of CPS by implementing online monitoring technique. For this aim, firstly the temporal specifications should be expressed in a formal language, temporal logic, and then synthesised on a monitoring equipment. TTL, as a formalism for expressing CPS temporal behavior, can express timing constraints commonly used in CPS including **level-based** timing constraints where Boolean predicates are constructed from analog signals. For example, the following expression defines a level-based timing constraint: "*whenever the value of signal X remains positive for 1 second, signal Y should become greater than 2V within 5 seconds*". It can be expressed as: $\Box(\Box_{[0,1]}(X > 0v) \rightarrow \Diamond_{[0,5]}(Y > 2v))$.

TTL also allows for expressing **event-based** timing constraints which are specified to constrain the relations between the occurrence time of events e.g. the latency between two events should be less than 2 seconds when the acceptable tolerance is 100 microseconds. $\mathcal{L}(\phi_1, \phi_2, 0.0001s) > 2s$.

For online monitoring of either level-based or event-based timing constraint, TMA firstly converts analog signals to discrete Boolean signals similar to [Maler and other, 2013] by extracting the timestamp of each transition, from *true* to *false* or vice versa.

The sets of rising edges Γ_r and falling edges Γ_f for a Boolean signal ψ are defined as below:

$$\Gamma_r = \{t^{\psi}_{r_1}, ..., t^{\psi}_{r_n}\}$$

and

$$\Gamma_f = \{t^{\psi}_{f_1}, ..., t^{\psi}_{f_n}\}$$

where $t^{\psi}_{r_i}$ and $t^{\psi}_{f_i}$ are the timestamps for the i^{th} rising and falling edge on ψ, respectively. Figure 5.1.a depicts a Boolean signal ψ, which is created when the signal $s(t)$ becomes greater than function $f(t)$. After threshold crossing, the Boolean signal is described by $t^{\psi}_{r_i}$ and $t^{\psi}_{f_i}$, $(i = 1, ..., n)$. Now, ψ is represented as a tuple consisting of an initial state (ψ_{init}), a set of rising edges (Γ_r) and falling edges (Γ_f): $\psi = (\psi_{init}, \Gamma_r, \Gamma_f)$. Note that this is a reversible operation and the Boolean signal can be reconstructed from the initial state ψ_{init} and sets of timestamps for rising and falling edges.

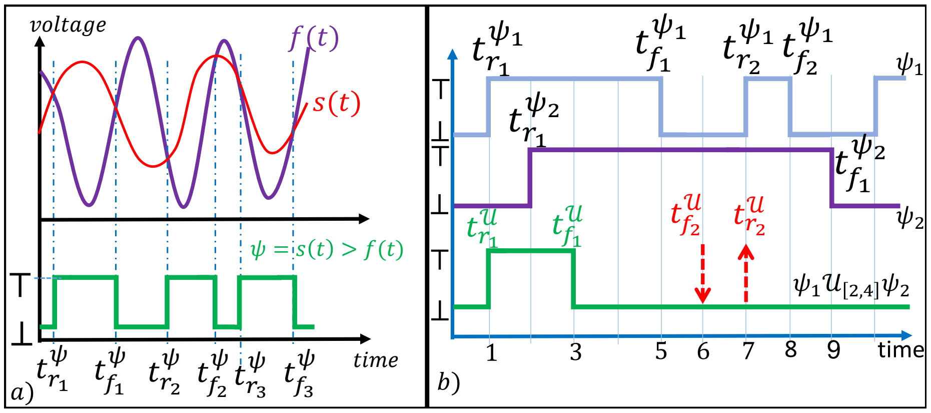

voltage
$f(t)$
$s(t)$
$\psi = s(t) > f(t)$
$\top$
$\bot$
$t^{\psi}_{r_1}$ $t^{\psi}_{f_1}$ $t^{\psi}_{r_2}$ $t^{\psi}_{f_2}$ $t^{\psi}_{r_3}$ $t^{\psi}_{f_3}$
time
a)
$t^{\psi_1}_{r_1}$ $t^{\psi_1}_{f_1}$ $t^{\psi_1}_{r_2}$ $t^{\psi_1}_{f_2}$
ψ_1
$t^{\psi_2}_{r_1}$
$t^{\psi_2}_{f_1}$
ψ_2
$t^{\mathcal{U}}_{r_1}$ $t^{\mathcal{U}}_{f_1}$ $t^{\mathcal{U}}_{f_2}$ $t^{\mathcal{U}}_{r_2}$
$\psi_1 \mathcal{U}_{[2,4]} \psi_2$
1 3 5 6 7 8 9 time
b)

Figure 5.1: a) Generating a Boolean Signal from Analog Signal, b) Calculating Until Operator in TMA.

By applying differentiate operator on a Boolean signal ψ ($\varphi = \bowtie \psi$), event signal is generated (φ). φ is *true* for a short period (sampling time) and *false* otherwise. Since this operator provides the event set, Θ^{φ}, it contains just the timestamps showing the time of ***events*** (not rising and falling), $\Theta^{\varphi} = \{\theta_1^{\varphi}, ..., \theta_n^{\varphi}\}$.

5.1 TMA Algorithms for Level-based Temporal Specifications

After converting the real-value signals to Boolean and the extracting the corresponding timestamps, now it is possible to introduce the TMA algorithms. For level-based timing operators there are three major algorithms.

5.1.1 Globally Operator

$\Box_{[a,b]}\psi$ at time t is true if the value of ψ in the future interval $[t+a, t+b]$ is **Always** true. In order to monitor such statement by TMA, signal ψ should be firstly converted to a set of rising and falling timestamps ($\Gamma_r = \{t_{r_1}^{\psi}, ..., t_{r_n}^{\psi}\}$ and $\Gamma_f = \{t_{f_1}^{\psi}, ..., t_{f_n}^{\psi}\}$). Given a Boolean signal (ψ) expressed with a set of rising and falling edges (Γ_r^{ψ} and Γ_f^{ψ}), the set of rising and falling edges for $\Box_{[a,b]}\psi$ ($\Gamma_r^{\Box}$ and $\Gamma_f^{\Box}$) are updated by running Algorithm 1.

The new $\Box_{[a,b]}\psi$ rising and falling edges are computed based on the most recent t_r^{ψ} (expressed as the current rising edge timestamp on ψ), t_f^{ψ} (expressed as the current falling edge timestamp on ψ) as well as the values of a and b. The computed rising and falling edges are only added to $\Gamma_r^{\Box}$ and $\Gamma_f^{\Box}$ if their timestamps for the rising edge is less than that of the falling edge. A pair of timestamps appended to $\Gamma_r^{\Box}$ and $\Gamma_f^{\Box}$ signifies that there is a new valid interval where the constraint, $\Box_{[a,b]}\psi$, was met.

5.1.2 Eventually Operator

$\Diamond_{[a,b]}\psi$ at time t is true if the value of ψ in the future interval $[t+a, t+b]$ becomes true **Eventually** at least for a moment. For monitoring $\Diamond_{[a,b]}\psi$ in TMA, a Boolean signal (ψ) expressed with, Γ_r^{ψ} and Γ_f^{ψ}, for every new pair of timestamps, the set of

Algorithm 1 Globally (t_r^ψ, t_f^ψ, a, b)

```
1:  $t_{r_i}^\Box = t_r^\psi - a$
2:  $t_{f_i}^\Box = t_f^\psi - b$
3:  if $t_{r_i}^\Box < 0$ then
4:      $t_{r_i}^\Box = 0$
5:  end if
6:  if $t_{f_i}^\Box < 0$ then
7:      $t_{f_i}^\Box = 0$
8:  end if
9:  if $t_{r_i}^\Box < t_{f_i}^\Box$ then
10:     $\Gamma_r^\Box = \Gamma_r^\Box + \{t_{r_i}^\Box\}$
11:     $\Gamma_f^\Box = \Gamma_f^\Box + \{t_{f_i}^\Box\}$
12: end if
```

rising and falling edges are updated by applying Algorithm 2. The new $\Diamond_{[a,b]}\psi$ rising and falling edges are computed upon the most recent t_r^ψ and t_f^ψ timestamps as well as the values of a and b. The calculated timestamps are only added to the set under the constraint; a rising edge must occur after the last falling edge. Also, if the last computed falling is in the range of new pulse, the last falling should be replaced with the new falling edge to append the last pulse on the result. A pair of timestamps appended to $\Gamma_r^\Diamond$ and $\Gamma_f^\Diamond$ signifies that there is a new valid interval where the constraint, $\Diamond_{[a,b]}\psi$, was met.

5.1.3 Until Operator

In $\psi_1\mathcal{U}_{[a,b]}\psi_2$, *Until* operator is applied on two Boolean signals ψ_1 and ψ_2 and $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is true at time t if ψ_2 is true at t', $t' \in [t+a, t+b]$, and ψ_1 is true

Algorithm 2 Eventually (t_r^{ψ}, t_f^{ψ}, a, b)

```
1: $t_{r_i}^{\Diamond} = t_r^{\psi} - b$
2: $t_{f_i}^{\Diamond} = t_f^{\psi} - a$
3: if $t_{r_i}^{\Diamond} < 0$ then
4:     $t_{r_i}^{\Diamond} = 0$
5: end if
6: if $t_{f_i}^{\Diamond} < 0$ then
7:     $t_{f_i}^{\Diamond} = 0$
8: end if
9: if $t_{f_{i-1}}^{\Diamond} < t_{r_i}^{\Diamond}$ and $t_{r_i}^{\Diamond} < t_{r_f}^{\Diamond}$ then
10:    $\Gamma_r^{\Diamond} = \Gamma_r^{\Diamond} + \{t_{r_i}^{\Diamond}\}$
11:    $\Gamma_f^{\Diamond} = \Gamma_f^{\Diamond} + \{t_{f_i}^{\Diamond}\}$
12: end if
13: if $t_{r_i}^{\Diamond} <= t_{f_{i-1}}^{\Diamond}$ and $t_{f_{i-1}}^{\Diamond} < t_{f_i}^{\Diamond}$ then
14:    $\Gamma_f^{\Diamond} = \Gamma_f^{\Diamond} - \{t_{f_{i-1}}^{\Diamond}\}$
15:    $\Gamma_f^{\Diamond} = \Gamma_f^{\Diamond} + \{t_{f_i}^{\Diamond}\}$
16: end if
```

continuously from t to t'. In order to monitor *Until* given two Boolean signals, ψ_1 and ψ_2, with new rising and falling edges $t_r^{\psi_1}$, $t_r^{\psi_2}$, $t_f^{\psi_1}$ and $t_f^{\psi_2}$, the set of rising and falling edges for $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ ($\Gamma_r^{\mathcal{U}}$ and $\Gamma_f^{\mathcal{U}}$) are updated by Algorithm 3 with the incoming pairs of timestamps. The new rising and falling edges for *Until* are computed in the first 2 lines.

Algorithm 3 Until ($t_r^{\psi_1}$, $t_r^{\psi_2}$, $t_f^{\psi_1}$, $t_f^{\psi_2}, a, b$)

```
1: $t_{r_i}^{\mathcal{U}} = max(t_r^{\psi_1}, t_r^{\psi_2} - b)$
2: $t_{f_i}^{\mathcal{U}} = min(t_f^{\psi_1}, t_f^{\psi_2}) - a$
3: if $t_{r_i}^{\mathcal{U}} < 0$ then
4:     $t_{r_i}^{\mathcal{U}} = 0$
5: end if
6: if $t_{f_i}^{\mathcal{U}} < 0$ then
7:     $t_{f_i}^{\mathcal{U}} = 0$
8: end if
9: if $t_{f_{i-1}}^{\mathcal{U}} < t_{r_i}^{\mathcal{U}}$ and $t_{r_i}^{\mathcal{U}} < t_{f_i}^{\mathcal{U}}$ and $t_{r_i}^{\mathcal{U}} < t_{r_f}^{\mathcal{U}}$ then
10:     $\Gamma_r^{\mathcal{U}} = \Gamma_r^{\mathcal{U}} + \{t_{r_i}^{\mathcal{U}}\}$
11:     $\Gamma_f^{\mathcal{U}} = \Gamma_f^{\mathcal{U}} + \{t_{f_i}^{\mathcal{U}}\}$
12: end if
13: if $t_{r_i}^{\mathcal{U}} <= t_{f_{i-1}}^{\mathcal{U}}$ and $t_{f_{i-1}}^{\mathcal{U}} < t_{f_i}^{\mathcal{U}}$ then
14:     $\Gamma_f^{\mathcal{U}} = \Gamma_f^{\mathcal{U}} - \{t_{f_{i-1}}^{\mathcal{U}}\}$
15:     $\Gamma_f^{\mathcal{U}} = \Gamma_f^{\mathcal{U}} + \{t_{f_i}^{\mathcal{U}}\}$
16: end if
```

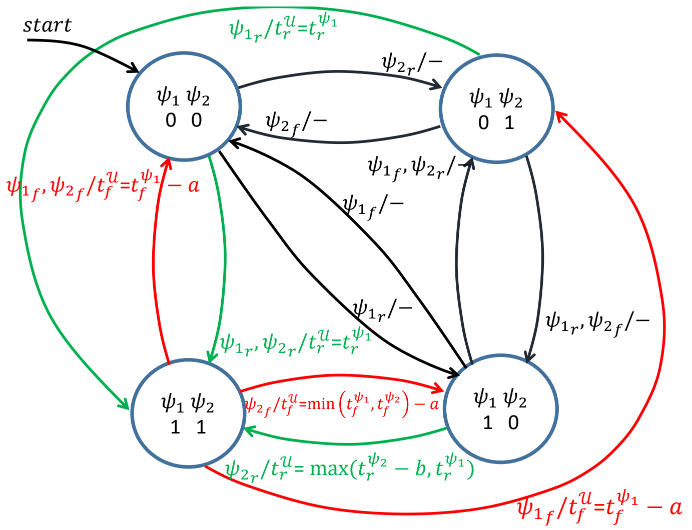

start
$\psi_{1_r}/t_r^{\mathcal{U}}=t_r^{\psi_1}$
$\psi_{2_r}/-$
$\psi_1\ \psi_2$ 0 0
$\psi_1\ \psi_2$ 0 1
$\psi_{2_f}/-$
$\psi_{1_f}, \psi_{2_r}/-$
$\psi_{1_f}, \psi_{2_f}/t_f^{\mathcal{U}}=t_f^{\psi_1} - a$
$\psi_{1_f}/-$
$\psi_{1_r}/-$
$\psi_{1_r}, \psi_{2_f}/-$
$\psi_{1_r}, \psi_{2_r}/t_r^{\mathcal{U}}=t_r^{\psi_1}$
$\psi_1\ \psi_2$ 1 1
$\psi_{2_f}/t_f^{\mathcal{U}}=\min(t_f^{\psi_1}, t_f^{\psi_2}) - a$
$\psi_1\ \psi_2$ 1 0
$\psi_{2_r}/t_r^{\mathcal{U}}= \max(t_r^{\psi_2} - b, t_r^{\psi_1})$
$\psi_{1_f}/t_f^{\mathcal{U}}=t_f^{\psi_1} - a$

Figure 5.2: FSM to Implement an *Until* Operator.

Starting at line 3, new edges are either appended or discarded, depending on whether or not they comply with the signals. For example, any negative time value and any set of edges with a falling happening before a corresponding rising edge indicate the constraint is not satisfied. Similarly, any edge with rising that comes before the falling edge of the previous set is discarded and the previous falling is replaced with the new falling since the last positive pulse should be extended to the new falling edge. A pair of timestamps appended to $\Gamma_r^{\mathcal{U}}$ and $\Gamma_f^{\mathcal{U}}$ signifies that there is a new valid interval where the constraint, $\psi_1\mathcal{U}_{[a,b]}\psi_2$, was met. As depicted in the *Until* example in Figure 5.1.b, $t_{r_1}^{\mathcal{U}} = max(1, 2-4) = 1$ and $t_{f_1}^{\mathcal{U}} = min(5, 9) - 2 = 3$. Since $t_{r_1}^{\mathcal{U}} < t_{f_1}^{\mathcal{U}}$ they can be used to update $\psi_1\mathcal{U}_{[2,4]}\psi_2$ by being appended to $\Gamma_r^{\mathcal{U}}$ and $\Gamma_f^{\mathcal{U}}$. The potential $\psi_1\mathcal{U}_{[2,4]}\psi_2$ rising and falling edges obtained from the second pulse of ψ_1 are then computed as follows: $t_{r_2}^{\mathcal{U}} = max(7, 2-4) = 7$ and $t_{f_2}^{\mathcal{U}} = min(8, 9) - 2 = 6$. Since $t_{f_2}^{\mathcal{U}} \leq t_{r_2}^{\mathcal{U}}$ they must be disregarded rather than appended to $\Gamma_r^{\mathcal{U}}$ and $\Gamma_f^{\mathcal{U}}$. This concludes that $\mathcal{U}_{[2,4]}$, were met in the interval from time $t = 1$ to $t = 3$, when the first pulse of ψ_1 must hold until the rising event on ψ_2 is true at some time step between a and b [1] . The Finite State Machine (FSM) in figure 5.2, calculates the result of *Until* operator with just four states (two bits).

5.2 TMA Algorithms for Event-based Temporal Specifications

For demonstrating how TMA monitors the event-based temporal specifications, *Latency* and *Simultaneity* operators are explained in this section where the rest of the TTL operator are monitored in a similar way. However, it is worth mentioning that in the diagrams for calculating *Latency* and *Simultaneity* in figures 5.3 and 5.4 the measurement errors are not showed. This is because the figures are depicted to make a better understanding of the method for the reader. Intuitively, chapter 3

[1]In the calculations for $\psi_1\mathcal{U}_{[a,b]}\psi_2$ operator,just overlapped pulses on ψ_1 and ψ_2 are considered.

and 4 represented all required calculations for measurement errors and user-defined tolerance values.

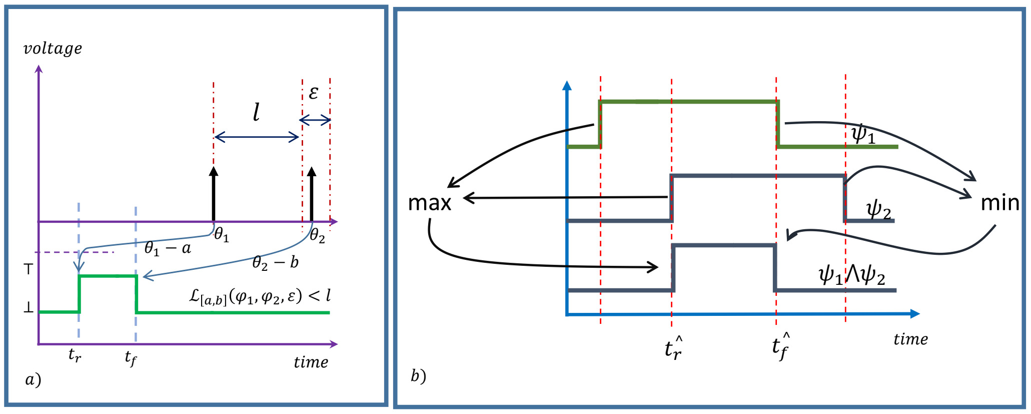


Figure 5.3: a) Calculating of *Latency* Constraint, b) Calculating *AND* Gate Using Timestamps.

5.2.1 Latency Constraint

A *Latency* constraint specifies the time difference between the occurrence of two events. A simple example of a latency constraint is the minimum, maximum or exact time interval between two events, denoted as follows: $\mathcal{L}(\varphi_1, \varphi_2, \varepsilon) \bigtriangledown l$ where $\bigtriangledown \in \{>, <, =\}$. The test code generation takes as input two events (φ_1 and φ_2) and compares the difference between the event timestamps with a real number l. Since the signals are singletons,the sets of Θ^{φ_1} and Θ^{φ_2} each contains only one element. Hence, whenever event Θ^{φ_2} is received, the latency can be calculated. The latency constraint evaluation is comprised of two steps: (1) calculating the delay Δt between two timestamps, $\theta_1^{\varphi_1}$, and $\theta_1^{\varphi_2}$ ($\Delta t = \theta_1^{\varphi_2} - \theta_1^{\varphi_1}$), and (2) comparing Δt with l. If $((\Delta t \bigtriangledown l) = \top)$ then the rising and falling edges of result are: $t_r^{\mathcal{L}} = \theta_1^{\varphi_2} - a$, $t_f^{\mathcal{L}} = \theta_1^{\varphi_1} - b$. As the example shows in figure 5.3.a, the predicate is $\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l$, the maximum *Latency*. Although the time difference between timestamps of events θ_1 and θ_2 (the corresponding events for ϕ_1 and ϕ_2) is a little bit more than l, since it does not exceed $l + \varepsilon$, the result is true from $\theta_1 - a$ to $\theta_2 - b$.

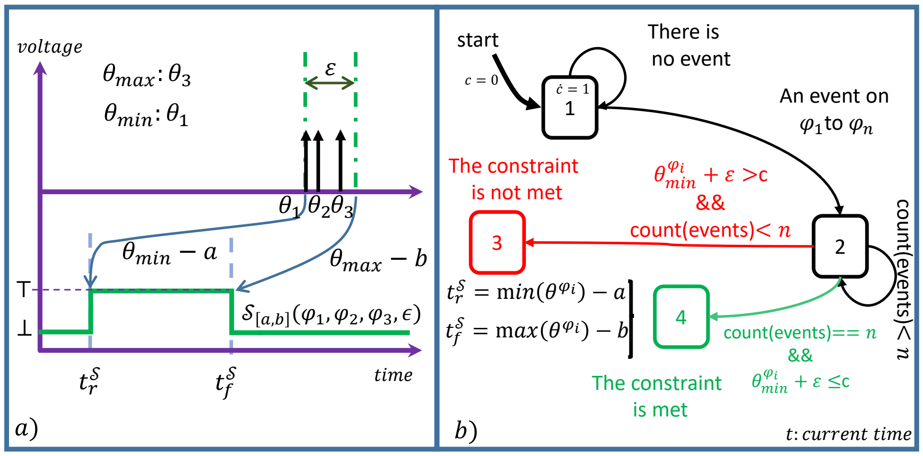

voltage
θ_{max}: θ_3
θ_{min}: θ_1
ε
θ_1 θ_2 θ_3
$\theta_{min} - a$
$\theta_{max} - b$
⊤
⊥
$\mathcal{S}_{[a,b]}(\varphi_1, \varphi_2, \varphi_3, \epsilon)$
$t_r^{\mathcal{S}}$
$t_f^{\mathcal{S}}$
time
a)
start
$c = 0$
$\dot{c} = 1$
1
There is no event
An event on φ_1 to φ_n
2
count(events)< n
The constraint is not met
$\theta_{min}^{\varphi_i} + \varepsilon >$c && count(events)< n
3
$t_r^{\mathcal{S}} = \min(\theta^{\varphi_i}) - a$
$t_f^{\mathcal{S}} = \max(\theta^{\varphi_i}) - b$
4
count(events)== n && $\theta_{min}^{\varphi_i} + \varepsilon \leq$c
The constraint is met
b)
t: current time

Figure 5.4: a) Calculation of *Simultaneity* Constraint b) The Timed-automata to Calculate Simultaneity Constraint.

5.2.2 *Simultaneity Constraint*

To determine the satisfiability of the *Simultaneity* constraint, the point in time where a set of events have occurred within a time tolerance of ε is evaluated. Figure 5.4.a) shows the example of three events occurring within ε so that the constraint is met between $\theta_{min} - a$ and $\theta_{max} - b$. A timed-automaton is utilized to evaluate this timing constraint. As figure 5.4.b) demonstrates, if the timed-automaton detects n events in ε duration, the constraint is evaluated to true.

5.2.3 *Untimed Operators' Computation Using Timestamps*

As it is discussed earlier, the benefit of the proposed method (TMA) is due to its ability to replace the history of signals with their latest transition timestamps. However, it is required to have the unique framework for all possible operators. Ordinary logic operators like And, Or, Implies or so on, just operate on Boolean signals and produce Boolean results as well [Moslem Didehban and Mehrabian, 2108]. Since the evaluation in TMA is done using timestamps, there should be a way to evaluate ordinary logic operators accepting timestamps. As an example, an **And** calculation approach is depicted in figure 5.3.b) where it calculates $\psi_1 \wedge \psi_2$. The corresponding FSM for *AND* gate is in figure 5.5. At the beginning, both ψ_1 and ψ_2 are false ($\psi_1 = 0$ and $\psi_2 = 0$). If the FSM receives a rising on ψ_1, ψ_{1_r}, the states of the predicates are $\psi_1 = 1$ and $\psi_2 = 0$. Whenever a rising is detected on ψ_2 as well, now the state of Boolean signals are *true* ($\psi_1 = 1$ and $\psi_2 = 1$). Therefore, the algorithm should produce the output which is *true* and it would be a rising edge timestamp on the output signal (the bottom green arrow). The rising edge timestamp is the same as the rising edge of ψ_2, $t_r^{\wedge} = t_r^{\psi_2}$ because only at that time both ψ_1 and ψ_2 had had their own rising timestamps.

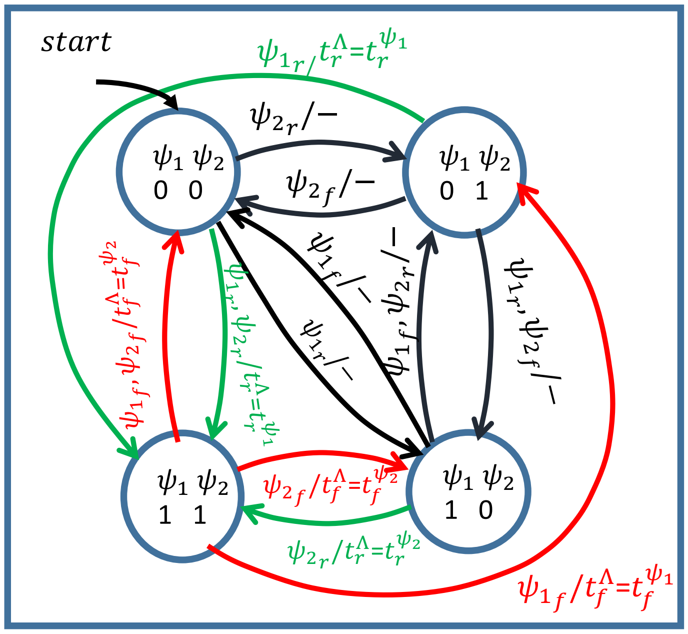

start
$\psi_{1_r}/t_r^{\wedge}=t_r^{\psi_1}$
$\psi_{2_r}/-$
$\psi_1\ \psi_2$ 0 0
$\psi_1\ \psi_2$ 0 1
$\psi_{2_f}/-$
$\psi_{1_f}/-$
$\psi_{1_f}, \psi_{2_r}/-$
$\psi_{1_r}, \psi_{2_f}/-$
$\psi_{1_r}/-$
$\psi_{1_f}, \psi_{2_f}/t_f^{\wedge}=t_f^{\psi_2}$
$\psi_{1_r}, \psi_{2_r}/t_r^{\wedge}=t_r^{\psi_1}$
$\psi_1\ \psi_2$ 1 1
$\psi_1\ \psi_2$ 1 0
$\psi_{2_f}/t_f^{\wedge}=t_f^{\psi_2}$
$\psi_{2_r}/t_r^{\wedge}=t_r^{\psi_2}$
$\psi_{1_f}/t_f^{\wedge}=t_f^{\psi_1}$

Figure 5.5: The FSM for Calculating AND Gate Using Timestamps.

5.3 Time Testing Methodology

In this section, a methodology is presented to automate the monitoring of CPS timing constraints that can be implemented on commercially available platforms. In this process, there are three entities: 1) System Under Test (SUT), 2) TTL statements that specify the timing requirements, and 3) a measurement system that monitors the SUT's timing specifications.

5.3.1 Methodology Steps

The proposed methodology has five steps to perform the testing on a CPS as follows:

1- Making TTL Parse Tree

In order to produce the parse tree from the TTL statement, all temporal constraints should be written as a string, and all operators should be separated by parentheses. Then, the expression string is converted into a list of tokens. In each step, one token is taken, then the rules are applied until all tokens are applied on the tree (a non-binary tree). After taking the last token, the parse tree is created. In parsing a TTL statement, there are three different types of tokens:

1. Operators
   - Temporal operators ($\mathcal{U}$, $\Box$, $\Diamond$, $\mathcal{L}$, $\mathcal{S}$, $\mathcal{C}$, $\mathcal{F}$, $\mathcal{P}$, and $\bowtie$)
   - Boolean operators ($\neg$, $\wedge$ and $\Rightarrow$)
   - Comparison operators ($>$, $<$, $=$, $\nearrow$ and $\searrow$)
2. Operands
   - Signals
   - Real numbers
3. Separators
   - "(", ")", "$\langle$" and "$\rangle$"

Table 5.1: The Rules to Create a Parse Tree from a TTL Statement

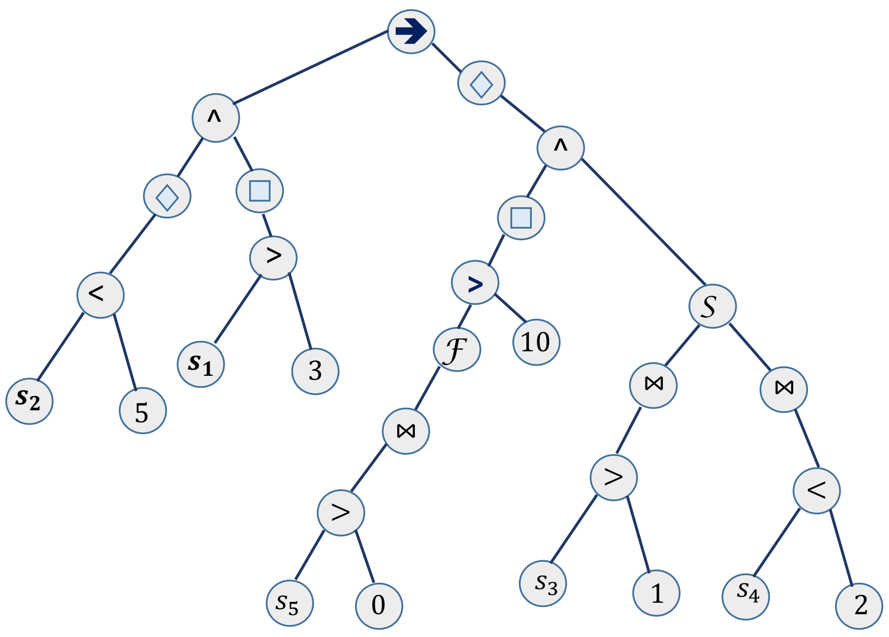
Current Token	Next Token	IsEmpty(G)	IsNotEmpty(G)
“(” or “⟨”	X	Create a new node	Create a right child
Operands	X	Create a right node and assign the operand to it	
All operators except “↗” and “↘”	X	Assign the operator to the current node	Travel upward until reach an empty node and assign the operator to it. If no empty node is found, create a parent node for the root, assign the operator to it and goto the new root
“,”	is not “⟨”	goto the parent node	
“,”	is “⟨”	goto the parent node and create an empty right child node for it.	
“)”	X	goto the parent node	
“⟩”	X	goto the parent node and assign “⋈” to it	
“↗”	X	assign “>” to the current node	
“↘”	X	assign “<” to the current node	

The right child and left child illustrate the ordering between the children of a node. The rules to parse a TTL statement are in table 5.1. In figure 5.6, a TTL statement is converted into its corresponding parse tree by the algorithm.

2- Creating the Block Diagram

Each node in the parse tree corresponds to a computing block in the block diagram except for leaves which are monitored signals and thresholds (figure 5.7).

Each block produces the output for its parent based on the timestamps it receives. *Chronological* and *Simultaneity* blocks produce Boolean output signal to show the exact time at which signals meet their constraints. The outputs of *Latency*, *Frequency* and *Phase* blocks are natural numbers and after comparing with a threshold and applying the differentiate operator they can be converted into signal events to be used in nested temporal operators.



$$\Big(\big(\big(\Box_{[0,5]}\,(s_1 > 3)\big) \wedge \big(\Diamond_{[0,5]}\,(s_2 < 5)\big)\big) \Rightarrow \big(\Diamond_{[0,5]}\,\big(\big(\Box_{[0,2]}\,\big(\big(\mathcal{F}\,(\langle s_5, 0, \nearrow\rangle)\big) > 10\big)\big) \wedge \big(\mathcal{S}\,(\langle s_3, 1, \nearrow\rangle, \langle s_4, 2, \searrow\rangle)\big)\big)\big)\Big)$$

Figure 5.6: Generated Parse Tree for a TTL Statement.

3- Creating the Physical Connection

After determining the list of monitored signals, an appropriately isolated data acquisition device should be chosen to measure the signals without changing the functionality of the CPS. For instance, the input impedance of the acquisition device should be high enough so that the system does not experience any voltage drop. Appropriately shielded cables should be used in order to avoid interference between signals, especially in high-frequency applications. Similarly, we need to use interface circuits like optocouplers to improve the isolation when isolation of the acquisition device is not high enough. The specifications of the monitoring devices, the way to make the connections between SUT and monitoring equipment and the architecture for distributed monitoring are discussed in chapter 6 in details.

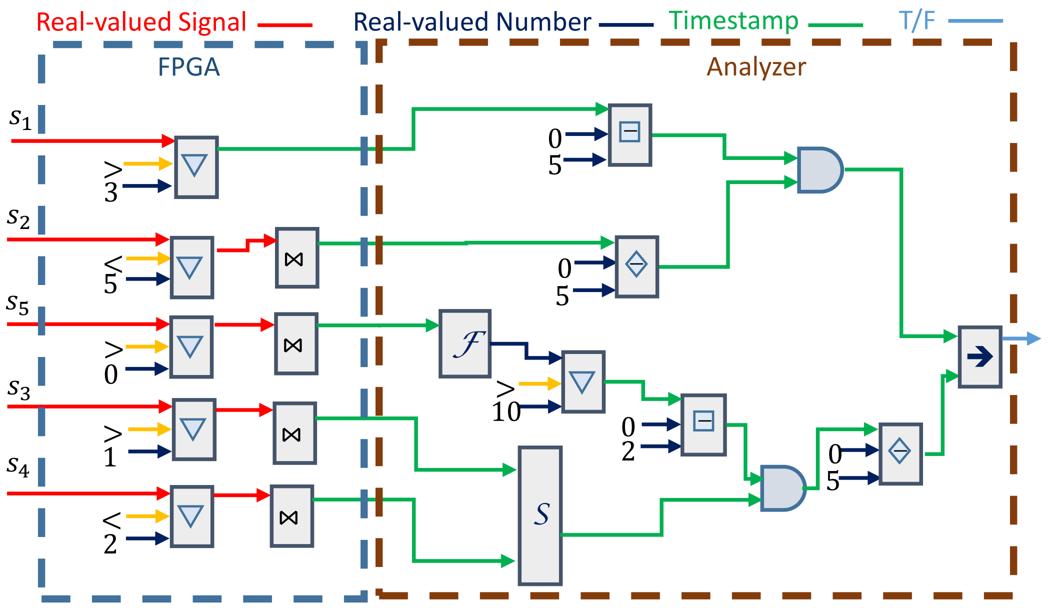
4- Signal Monitoring

In order to deal with timing constraints in TTL, real-valued signals should be represented as signal events based on the edge type (rising, falling). Then, using a reliable clock, signal events are converted to timestamps and stored in a database for either offline analysis or sent to an online testing application. TTL semantics in table 3.1 are expressed for continuous signals, where each signal event represents a single point in time. However, implementation on hardware platforms requires discretization of the continuous signal. It is assume that all signals are well-behaved such that there are no undetectable transient events between contiguous pairs of samples. This can be achieved by using appropriate Data Acquisition (DAQ) devices that have high sampling rates.

Time synchronization, absolution or relative, is necessary in measuring temporal properties of a distributed SUT. A distributed test device must be synchronized to ensure accurate, reproducible and repeatable measurement of the SUT based upon the temporal constraints evaluated. Providing timestamps based on a global time is an option in a system in which the occurrence of an event in absolute time matters. Otherwise, measuring the relative time between events is sufficient, and time testing can be implemented by the methodology regardless of accuracy to global time.

5- Constraint Evaluation

The result of a TTL statement is evaluated from the right-most block of the analyzer (figure 5.7). Past operators are used to specify a timing constraint over past time intervals as defined in [Maler and other, 2013, Jakšić*et al.*, 2015] since they should wait for future time to evaluate. For evaluating operators $\Diamond_{[a,b]}$ and $\Box_{[a,b]}$, the analyzer waits for b seconds and then starts evaluating the constraint. Therefore, the response



$$(((\Box_{[0,5]}(s_1 > 3)) \wedge (\Diamond_{[0,5]}(s_2 < 5))) \Rightarrow (\Diamond_{[0,5]}((\Box_{[0,2]}((\mathcal{F}(\langle s_5, 0, \nearrow\rangle)) > 10)) \wedge \mathcal{S}(\langle s_3, 1, \nearrow\rangle, \langle s_4, 2, \searrow\rangle))))$$

Figure 5.7: TTL Computing Blocks According to the Parse Tree in Figure 5.6.

at each instance (t) actually is corresponding to $t - b$. Moreover, online monitoring of a constraint that contains future operators with infinite time interval is obviously impossible (e.g. $\Box(x > 3)$). In order to solve this issue, the compiler modifies the operator to one with time interval, $[t_s, t_f]$ where t_s and t_f are start and finish time of the testing.

5.3.2 Methodology Capabilities

Since the methodology works based on the events' timestamps, it requires less memory for monitoring and is suitable for implementing future operators in online testing [2] . Moreover, the methodology completely fit distributed cyber-physical systems since it can evaluate the timing requirements when the agents are geographically posed in different locations. As long as all agents are synchronized and their clock drifts are less than a certain value it is possible to monitor remote agents. As an

[2]chapters 5 and 4.

example, assume the latency between two events ϕ_1 and ϕ_2 is a timing requirement but ϕ_1 occurs in $Agent_1$ and ϕ_2 is generated in $Agent_2$. $Agent_1$ and $Agent_2$ are two subsystem in a CPS that are geographically distributed. Since the method considers just the timestamps of ϕ_1 and ϕ_2 and it is not required to access the original signals, the calculation for latency is possible and can be conducted in one of two agents. Furthermore, since in distributed systems the synchronization error and the precision of event capturing devices matter, the methodology can support such issues by standardizing the monitoring testbed [3] .

The other capability of methodology is about using the output of TTL operators as input signals. For instance, for evaluating *Frequency*, *Latency* and *Phase* constraints, their outputs (a real-valued number) are compared with another number and provides a True/False signal. This True/False signal is represented by a set of rising and falling timestamps. As figure 5.7 shows, once the frequency of the signal s_5 is greater than 10 Hz, the output will be true. This Boolean signal is represented as two sets of timestamps containing rising and falling edge. Then, the output timestamps are passed to the globally operator($\Box$). The globally operator subtracts 2 s from every signal's falling edge timestamp. If the result is less than the previous signal's rising edge timestamp, the timestamp is fully removed.

In addition to aforementioned benefits, the methodology has the capability to be implemented on FPGA boards which are fast, reliable and low-cost. The acquired timestamps can be analyzed on the FPGA itself, transferred to a machine with a higher performance to process, or stored in a database when the CPS is distributed for offline analysis.

[3]chapter 6

Chapter 6

A METHOD TO QUALIFY TESTBEDS TO VERIFY TIMING BEHAVIOR OF CPS

This chapter is an effort towards standardizing the process of testing the timing properties of CPS where a design of a testbed is outlined. The testbed can be used to test the CPS to check if all the timing constraints are being met or not in a systematic and correct manner to enable correct-by-construction (CbC) synthesis of the testbed. The testbed – like the distributed CPS it is trying to test – is also a distributed CPS, with each node (of the testbed) monitoring the required signals from the CPS node. Hardware timestamping and IEEE 1588 Precision Time Protocol (PTP) synchronization of the clocks among the CPS components provides observations at the same timescale through vast geographies, without losing accuracy with time. In this chapter, there is also a discussion about the key timing parameters of the testbed that will affect the time testing capability. In this regard, it studies the specifications that must be met by the testbed, such that the testbed can validate the timing constraints. Finally, the Correct-by-construction (CbC) timing testbed is applied to verify the timing constraints of two example distributed CPS.

6.1 Distributed testbed to evaluate timing behavior of (distributed) CPS

A systematic approach to test and verify the timing behavior of a distributed system is to monitor signals and events of the System Under Test (SUT) and timestamp events with a common testbed timebase within the specified precision and accuracy such that the required timing constraints over a distributed system can be properly verified. Figure 6.1 shows the structure of the proposed distributed testbed. Just like the CPS under test, the testbed itself is also a distributed CPS. Each node of the CPS is monitored by a (Data Acquisition) DAQ platform – a programmable test and measurement device with analog and digital inputs and outputs. There are k distributed testbed nodes that communicate using a network link. All nodes are synchronized with a clock reference to have the same notion of time. There exists a database logging the time-stamped events from the distributed test and measurement system for further analysis.

Each testbed node has two major components:

- DAQ platform
- Operating System (OS)

The DAQ platform of each node monitors the signals and timestamps of the events of interest using a time-synchronized internal clock. The monotored signals must be made observable by design – this is an aspect of Design For Testability (DFT). Each test and measurement node is synchronized to a local clock reference – a Global Navigation Satellite System (GNSS) traceable reference time source for example – and distributed using the Precision Time Protocol (PTP). The event timestamps are sent to the OS, which in turn, sends them all the workstations data, that loads them into a database. The database of time-stamped events enables automated analysis to determine if the timing constraints of the distributed CPS are met or not. Whether a testbed can validate a timing constraint or not, depends on the design parameters of the testbed. The most important design parameters of the distributed testbed that affect the errors in the timing measurements are described below.

Whether a testbed can validate a timing constraints or not, depends on the design parameters of the testbed. The most important design parameters of the distributed testbed that affect the errors in the timing measurements are described below.

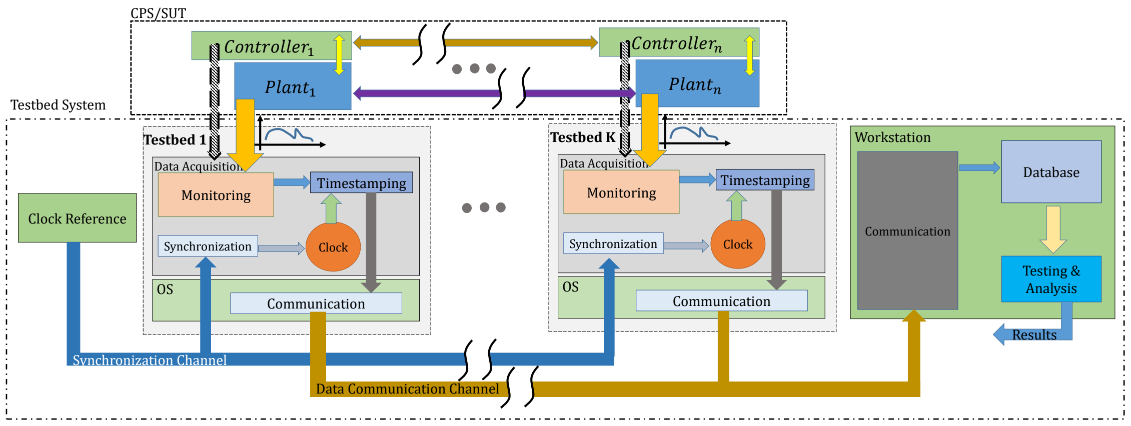


Figure 6.1: Time Testing Structure Diagram.

6.2 Analog to Digital Converter (ADC) parameters

The testbed monitors all signals by sampling them because they should be digitalized to use in cyber side. This is done by Analog to Digital Converters (ADCs) on the probes.

6.2.1 ADC Sampling Rate

The sampling rate of an ADC, f_s, is expressed as samples per second, or Hertz (Hz). In order to be able to monitor a signal correctly, the sampling rate must be sufficiently high to capture the fastest observable dynamics of interest in the signal. Suppose we intend to find out the time at which a signal rises above 3.4V. Figure 6.2 shows this signal, monitored with two different sampling rates. On the left with sampling rate of $f_s = 1$ kHz, the threshold crossing time of the signal is detected as $t = 1$ ms. However, on the right, with sampling rate of $f_s = 0.5$ kHz, the threshold crossing time of the signal is detected as $t = 2$ ms.

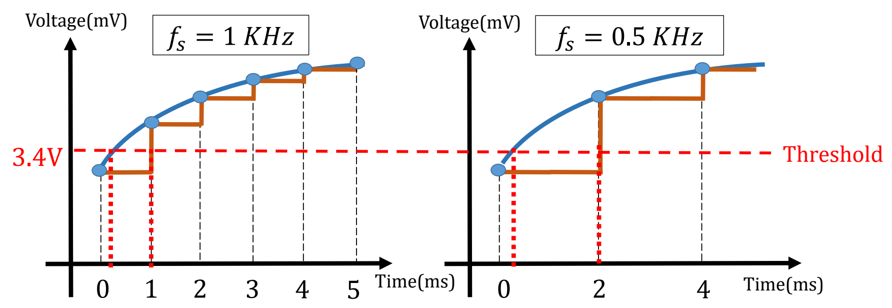


Figure 6.2: A Digitized Analog Signal at Two Different Sampling Rates.

6.2.2 ADC Resolution

Since an ADC converts the voltage signal into digitized sampled events, the accuracy of measurement is also limited by the number of bits used to express the sampled value, $nbits_{ADC}$, and the voltage range of the ADC, VR_{ADC}. An n-bit ADC can represent 2^n values. A 12-bit ADC that measures the range of 0 V to 5 V has steps of ≈ 1 mV. The precision of the ADC is defined in terms of resolution of the ADC, or V_{ADC} can be calculated as: $V_{ADC} = \frac{VR_{ADC}}{2^{nbits_{ADC}}}$. The resolution of the ADC can affect the time at which the monitoring device detects an event on a signal. Figure 6.3 illustrates the conversion of an analog signal to digital samples with various resolutions, VR_{ADC} of 5 V. If the user wants to detect the time when a signal rises above 4 V, then in the left diagram, with $nbits_{ADC} = 12$, the time at which the threshold crossing is detected is $t = 3$ ms, while in the right diagram, with $nbits_{ADC} = 11$, the time at which the threshold crossing is detected is $t = 4$ ms.

6.3 Input Impedance

Wiring a signal to a DAQ device adds a load to the CPS circuit under test, which causes a change in the shape of the monitored signal. For pure resistive loads, this change is a simple voltage drop while for general loads, the shape of the monitored

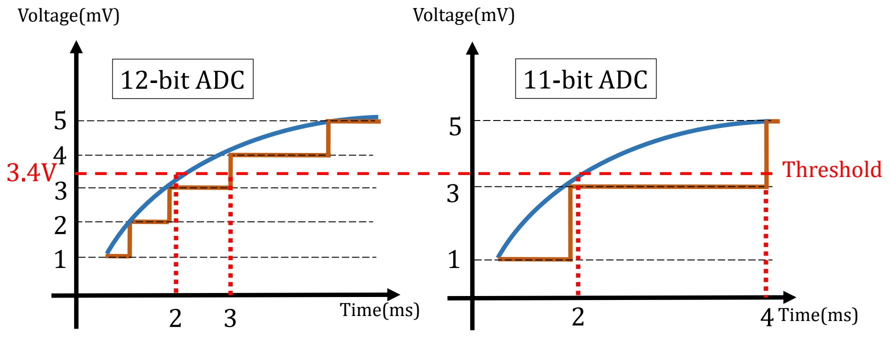


Figure 6.3: An Analog Signal Sampled Using Two Different ADCs That Have the same range and different resolutions.

signal is changed based on the equivalent resistance and reactance of the measuring device (including capacitance effect of the cables) and the SUT. As a result, based on the rate of change in the value of the signal, the measurements of the signal may be delayed or its amplitude may be attenuated. Input impedance, Z_{in}, is defined as the CPS equivalent circuit from the terminal connected to the testbed. The test and measurement device must have a sufficiently high input impedance to minimize perturbation of the measurement process on the signal. Figure 6.4.a shows the monitored signal perturbed by the loading effect of wiring the measurement device to the SUT. The threshold detection time of the original signal is after the threshold detection time of the monitored signal. The parts of b and c of the figure demonstrate the same potential problem for periodic and/or arbitrary signals.

6.4 Clock Parameters

A clock's fractional frequency offset is defined as $f_{clock} = \frac{f_{inst}-f_0}{f_0}$, where f_{inst} is the instantaneous clock frequency, and f_0 is the nominal clock frequency. Thus, this is the unitless instantaneous fractional offset from the nominal frequency of an oscillator [nis, 2017]. Environmental conditions such as voltage and temperature variations or mechanical vibrations, can affect the rate at which an oscillator runs.

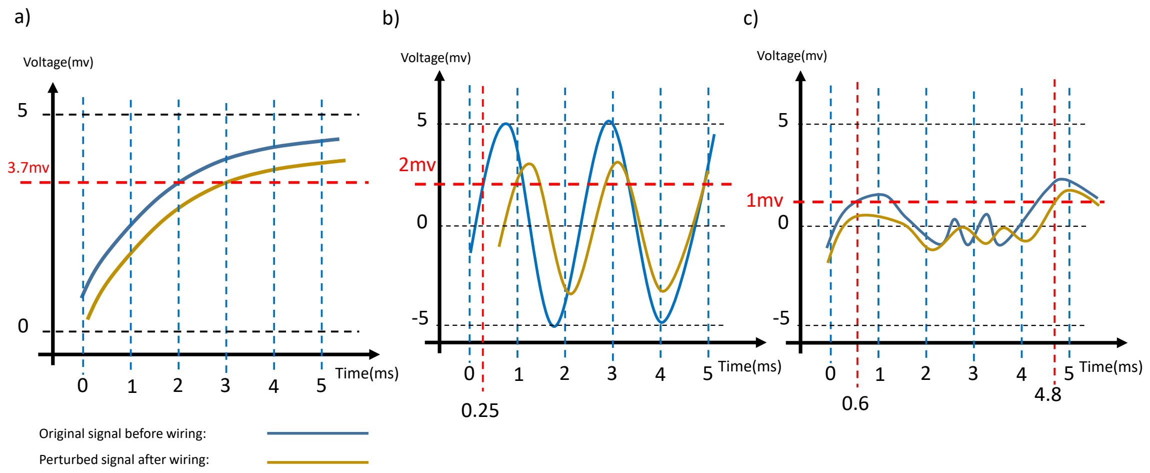


Figure 6.4: Voltage Drop on Different Signals Connected to a Resistive Load.

6.4.1 Clock Fractional Frequency Offset

Typically, the fractional frequency offset of a clock, f_{clock}, is expressed in Parts Per Million (PPM), indicating the maximum amount of error in one million time units. Thus, the time error after an elapsed time $t_{elapsed}$ due to a fractional frequency offset of f_{clock} is $t_{elapsed} \times f_{clock}$. For instance, a clock with 5 PPM error, has 5 μs error after 1 second, an error of about 0.5 s after a day, or about 2.5 min after a year.

Since all clocks deviate from each other, distributed clocks must be synchronized to a reference to have an agreement on time and have a unique and time notion. Synchronization protocols match the clock of a device to a reference clock. However, no synchronization protocol is perfect, and there is a synchronization error t_{sync}, that depends on several factors, including the number of bits used to represent the time, when the time stamping is done (e.g., in the hardware or in software), network jitter, network asymmetries delays, etc. [Eidson and Stanton, 2015]. The Network Time Protocol or NTP [Mills, 1992] can usually keep time synchronized to within tens of milliseconds over the public Internet ($t_{sync} \approx 10$ ms). The Precision Time Protocol, PTP [IEE, 2008], can provide time synchronization over a LAN with sub-microsecond accuracy. PTP with the White Rabbit[Lipiński et al., 2011] extension used for the CERN Large Hadron Collider, can synchronize to sub-nanosecond accuracy. For CPS distributed over a wide area with high precision and accuracy needs, GNSS (Global Navigation Satellite Systems) can provide 100 ns accuracy.

6.4.2 Clock Synchronization Rate

Another important parameter is the rate of synchronization, r_{sync}, which is the number of times per second (e.g., in units of Hz) that synchronization is performed. Every time we perform synchronization, the time offsets are within t_{sync} of each other. But from thereon, until the next synchronization, the clock times will move apart at the rate of f_{clock}, if the local clock uses the protocol to adjust its time but not its frequency. The worst-case clock offset, ϵ_{wcco}, while the system clock is synchronized via the time synchronization protocol in steady-state and while all other environmental conditions are stable, can be calculated as: $\epsilon_{wcco} = t_{sync} + \frac{f_{clock}}{r_{sync}}$. Note that the units are in time, since f_{clock} is unitless and the reciprocal of r_{sync} is in units of time.

Ideally, the testbed would have a GNSS receiver clock, where GNSS propagates time traceable to UTC (USNO) at an accuracy on the order of 100 ns. GNSS receivers can be installed into time servers and propagate traceable time through the network. The synchronization network can propagate the traceable reference time via PTP or NTP.

6.5 Testbed Capability Analysis

In order to determine whether timing behavior is verifiable by a given testbed, it is important to understand the sources of timing measurement uncertainty, described as ε in the timing constraint specifications.

Consider a distributed CPS, with a max latency constraint $\mathcal{L}(\phi_1, \phi_2, \varepsilon) < l$ for events ϕ_1 and ϕ_2, where ϕ_1 occurs on signal s_1 and ϕ_2 on s_2 respectively. These events are detected at different nodes of the CPS. The latency constraint states that, given t_1 as the occurrence of ϕ_1, the time at which ϕ_2 occurs should be less tha $t_1 + l + \varepsilon$. The testbed must capture the time at which an event occurs. However, the measured time will be erroneous. This can due to the several factors, including the sampling frequency f_s, the ADC resolution V_{ADC}, and the clock error, ε_{wcco}.

Consider an event described by the tuple $\phi_1 = \langle s_1, v_t, \nearrow \rangle$, marking the threshold v_t crossing of signal s_1 on a rising edge. Since the ADC output is a multiple of the supported resolution, the testbed may not be able to detect the exact point of the threshold crossing. Thus, the threshold value must be mapped to the nearest upper bound of the value. Since all sampled data are collected at known points in time (integer multiples of $\frac{1}{f_s}$), a threshold crossing is detected with a maximum error $\varepsilon_{ADC} = \frac{1}{f_s}$. Figure 6.5 illustrates the worst-case error $\frac{1}{f_s}$ in an example.

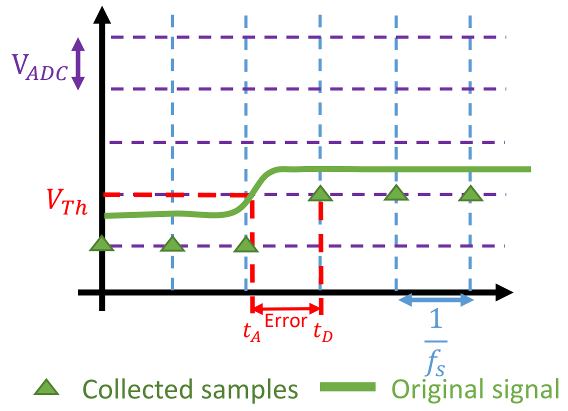


Figure 6.5: Worst-case Error between Actual Occurrence Time and Detection Time for an ADC with Sampling Frequency f_s.

Since all samples are timestamped using the local clock of the measurement system, clock synchronization error (ε_{wcco}) must be taken into account. Thus, the maximum time error between the actual event occurrence and the detected event occurrence is the sum of the ADC error and the clock synchronization error: $\varepsilon_{total} \leq \varepsilon_{wcco} + \varepsilon_{ADC}$.

Since there will be at most ε_{total} error in both the measurements of ϕ_1 and ϕ_2, then the testbed can confidently verify whether the exact latency constraint is being met or not. Other types of constraints (e.g., simultaneity, frequency, phase, etc.) are also expressed with a temporal error tolerance and one can similarly reason and verify the temporal behavior.

Chapter 7

APPLICATIONS AND EMPIRICAL EVALUATIONS

In order to demonstrate the usefulness of the proposed methods, four CPS applications were implemented and evaluated: i) flying paster, ii) an intersection manager for autonomous vehicles that manages the safe and efficient movement of autonomous vehicles through a traffic intersection (Robust Intersection Manager (RIM)), iii) simultaneous image capturing, and iv) synchrophasor. Firstly, the applications are introduced with their timing specifications in TTL. Then, the benefits of reasoning system are studied with applying the deduction system on the application's temporal specifications. In the next part, a monitoring system is proposed for each application using TMA and finally, the specifications of monitoring systems are examined to know whether they are good enough for monitoring.

7.1 Applications, Their Timing Specifications, and Modeling

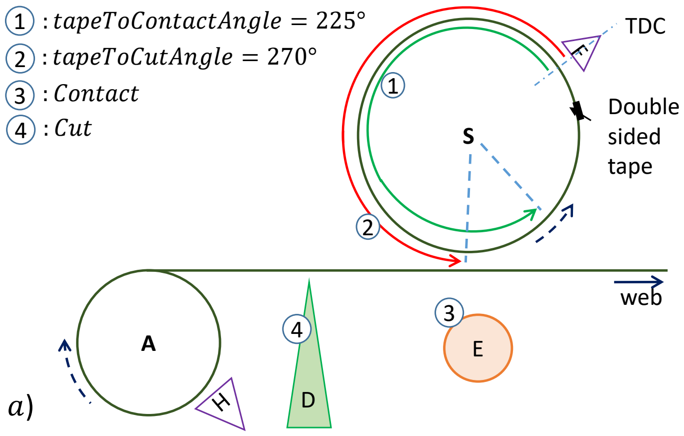
I this section, the applications used to show the capabilities of methods are introduced. In each part, the application description, its timing specifications and their expression in TTL, and how they have implemented are explained.

7.1.1 Flying Paster Application

A flying paster is part of a printing press, a distributed system enabling continuity of operation through the automatic exchange of an expiring paper roll with a new roll. Figure 7.1.a shows a schematic of the flying paster with the active roll A, which feeds the web. When the radius of paper in roll A is less than a given threshold, the roll is replaced by the spare roll S. The radius of the paper around roll A, (r_A), is measured by sensor H. When this radius falls below a given threshold, the *Approaching Out of Paper* (**AOP**) event (ϕ_{AOP}) is generated, which initiates the paper roll replacement process by starting the rotation of roll S. A strip of adhesive tape on paper roll S is used to attach the paper from roll S to roll A. The location of the tape is detected by sensor F, which creates the γ event (ϕ_γ). The frequency of the γ event is used to calculate the angular velocity ω_S of S. Once the linear velocities of roll S and A are equal, a **Match** event (ϕ_{Match}) is generated. Then, sensor F generates the event *Top Dead Center* (TDC) to indicate the detection of the tape. Two complete rotations of S after event **TDC**, the idler wheel E pushes the paper from roll A towards roll S, at which point the paper from roll S adheres to the outgoing paper from roll A. This event, ($\phi_{Contact}$), occurs after roll S performed two rotations plus 255 degrees, $tapeToContactAngle = 225°$. Immediately after it, the Cutter D cuts the paper from A. This is called the **Cut** event (ϕ_{Cut}), and occurs when roll S has two rotations plus $tapeToCutAngle = 270°$ after TDC.

Modeling of Flying Paster

A picture of the implementation of a scaled model of the flying paster is shown in figure 7.1.b. Rolls A and S in figure 7.1.a are implemented using two Hansen DC motors dialed ($0 - 360$ degree) disks, driven by two Arduino Mega2560 boards. Disks have a hole at *zero* degrees, which is detected by a photo-micro sensor. Photo-micro sensors implement the sensor H, and F and are installed next to the disks. The paper is modeled in software with the initial length of $125m$ and $0.05mm$ of thickness so that the initial diameter for both rolls is $9cm$ (radius of 4.5 cm). The **AoP** event is generated when the radius of A becomes less than $2.5cm$.



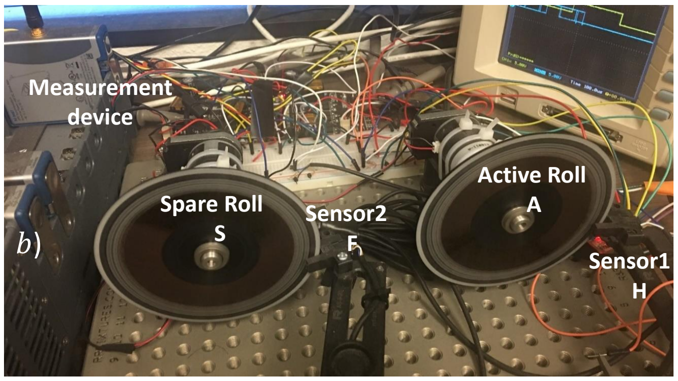


Figure 7.1: The Schematic of Flying Paster and its Modeling Implementation.

7.1.2 *Temporal Specifications of Flying Paster*

- $\mathcal{F}(\phi_\gamma, 0.005 \times \frac{v_A}{2\pi r_S}) = \frac{v_A}{2\pi r_A}$: The linear velocity of the paper (v_A) (measured by sensor F) of the active roll should be constant at $20m/s \pm 10^{-3}m/s$, otherwise, it cannot be fed to the printing press. It is known that $v_A = r_A \times \omega_A$ where r_A is the current paper radius and ω_A is the angular velocity of the roll A. The tolerable error for the velocity is 0.5 percent of the velocity on the active roll ($0.005 \times \frac{v_A}{2\pi r_S}$).

- $\mathcal{L}(\phi_{AOP}, \phi_{Match}, 10^{-4}s) < 6s$: The time interval from **AoP** to **Match** should be no more than 6 seconds with the maximum of $100\mu s$ ($10^{-4}s$) of uncertainty. Otherwise, the paper on the old roll will run out before the new paper can be attached.

- $\mathcal{C}(\phi_{Match}, \phi_{TDC}, 10^{-4}s)$ and $\mathcal{L}(\phi_{Match}, \phi_{TDC}, 10^{-4}s) < \frac{2\pi}{\omega_S}$: This captures the specification that **Match** happens before **TDC**, and that **TDC** happens before completing one full rotation from **Match**. Otherwise **Contact** and **Cut** do not work correctly.

- $\mathcal{C}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s)$: *Cut* event must occur after *Contact* event, otherwise, the paper from the active roll is cut before attaching the new paper. The allowed tolerance to detect this chronology is $100\mu s$ ($10^{-4}s$).

- $\mathcal{L}(\phi_{Contact}, \phi_{cut}, 10^{-4}s) < 1.5ms$: The delay between **Contact** and **Cut** should be less than $1.5ms$ not to have a late cut. The acceptable tolerance is $100\mu s$ ($10^{-4}s$).

- $\mathcal{L}(\phi_{TDC}, \phi_{contact}, 10^{-4}s) < \frac{4\pi}{\omega_S} + \frac{225}{\omega_S}$: This captures the specification that **Contact** must occur 2 rotations plus 225 degrees from the **TDC** event. Otherwise the two papers are not connected.

- $\mathcal{L}(\phi_{TDC}, \phi_{cut}, 10^{-4}s) > \frac{4\pi}{\omega_S} + \frac{270}{\omega_S}$, when tape is in 2 rotations plus 270° of TDC, **Cut** must fire. Otherwise, old paper is not cut in time.

7.2 Autonomous Intersection Manager (RIM) Application

The second case study is of an intersection manager (IM) for autonomous vehicles. Several designs of traffic intersections for autonomous vehicles has been proposed [Khayatian et al., 2020a, Dedinsky et al., 2019, Khayatian et al., 2020b, 2021, 2020c]. Robust Intersection Manager (RIM) [Khayatian *et al.*, 2018] is one of the recent frameworks for autonomous intersections that works in presence of variable round-trip delay in communication between the vehicle and the intersection, and is robust against model mismatches and external disturbances. Figure 7.2 depicts the RIM algorithm. The intersection operates in 4 phases: i) when an incoming vehicle reaches the synchronization line (SL), it sends a request to IM to synchronize its clock with the IM. ii) When the vehicle crosses the request line (RL), it requests a VoA and ToA – the velocity and time of arrival of the vehicle to the edge of intersection. iii) The vehicle receives the VoA and ToA at some time say t_θ, and then creates and follows an optimal trajectory to achieve the assigned VoA and ToA. iv) When the vehicle reaches the intersection line (IL), it cruises across the intersection at VoA. When a vehicle receives VoA and ToA, its controller targets to be at IL with velocity of VoA at ToA.

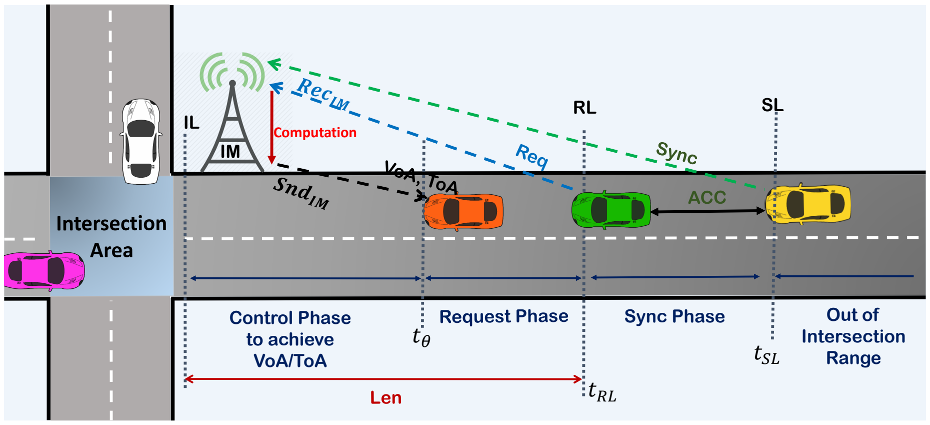


Figure 7.2: Robust Intersection Manager (RIM) Algorithm.

Modeling of RIM

The intersection manager case study consists of 6 autonomous vehicles and an intersection manager. Vehicles communicate with the IM through a wireless network. The length of each road connected to the intersection is 3m and the lane width is 1m. Each vehicle is built on the Traxxas chassis, which is 0.6m long and 0.2m wide. The RL and the SL are placed 2.5 meters and 3 meters away from IL. The intersection manager and vehicles are developed with ESP8266 microcontrollers. The Network Time Protocol (NTP) is used for clock synchronization and have $\delta = 8\mu s$ as the maximum synchronization error. The measurement error is replaced with synchronization error.

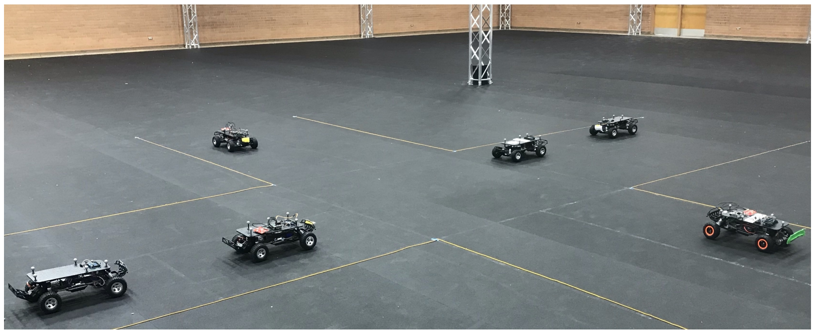


Figure 7.3: Autonomous Intersection Manager Using Traxxas Cars.

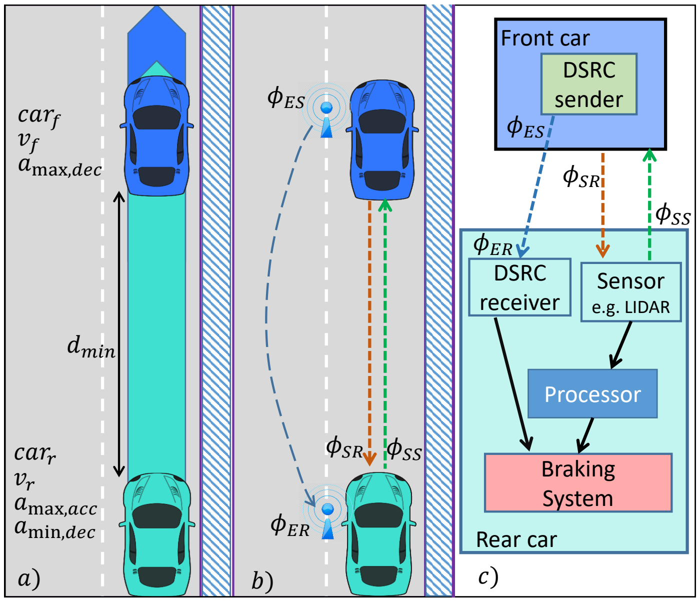


Figure 7.4: RSS Longitudinal Distance Rule and the Relations between the Events.

7.3 Applying Reasoning System on two Applications

A Scenario for RIM

In order to know the timing specifications for RIM, firstly we need to define a scenario. In RIM, Adaptive Cruise Control (ACC) is used to ensure front-back accident will not happen if a vehicle breaks suddenly. In ACC, the rear vehicle detects that the front vehicle applies the brake in two ways, i) receiving the sudden brake message through Dedicated Short-Range Communications (DSRC) or ii) the measured distance is less than a threshold.

The sensor measurements enable the vehicle to maintain a minimum distance from the front vehicle based on the worst-case network delay and response time of detection. The rear vehicle can detect an anomalous delay in the network by continuous monitoring of the received data from the front vehicle. If network delay is greater than a threshold, the minimum distance should be increased. This is because the rear vehicle maintains a safe distance by using a LIDAR or RADAR-based sensor, and increased network delays result in longer response times.

The Responsibility-Sensitive Safety (RSS) [Shai and other, 2017] provides a framework to determine the safe distance for a rear vehicle to follow in order to avoid an accident in the assumed worst-case scenario. The equation below shows the relation between minimum distance and response time, ρ (time between detecting a distance less than $d_{\min}$ to brake):

$$d_{\min} = \left(v_r \rho + \frac{1}{2} a_{\text{max,acc}} \rho^2 + \frac{(v_r + \rho a_{\text{max,acc}})^2}{2 a_{\text{min,dec}}} - \frac{v_f^2}{2 a_{\text{max,dec}}} \right) \tag{7.1}$$

where v_r and v_f are velocities of the rear and front vehicles, $a_{max,acc}$ is the maximum acceleration rate and, $a_{min,dec}$ and $a_{max,dec}$ are the minimum and maximum deceleration rates respectively. As figure 7.4.a shows two autonomous vehicles must have a minimum longitudinal distance, $d_{\min}$, before IL.

Timing Specifications of RIM

Based on the desired performance and safety of RIM, the following timing specifications are for RIM:

- $\mathcal{L}(\phi_\theta, \phi_{IL}, 10^{-3}s) > th$, the time between receiving VoA/ToA (ϕ_θ) and crossing IL (ϕ_{IL}) must be greater than a threshold (th) with $1ms$ of error, otherwise the vehicle may not be able to achieve its VoA/ToA, which may lead to an accident. th is vehicle-dependent and determined by the vehicle specifications.

- $\mathcal{L}(\phi_{RL}, \phi_{IL}, 10^{-3}s) > \frac{len}{V_{max}}$, if the vehicle drives with a velocity of more than the maximum allowed speed, it would enter the intersection sooner than ToA and collide with another car.

- $\mathcal{L}(\phi_{RL}, \phi_{IL}, 10^{-3}s) < \frac{len}{V_{min}}$, if the vehicle drives with a velocity of less than the minimum allowed speed, it would enter the intersection later than ToA and it would collide with other cars.

- $\mathcal{F}(\phi_S, 5Hz) < \rho$, in the rear vehicle's Adaptive Cruise Control System, periodicity between sending radar measurements (ϕ_S) and distance computations must be less than braking response time (ρ). Otherwise, it cannot act fast enough and would collide with the front car. The allowable tolerance in the frequency of ϕ_S is $5Hz$.

- $\mathcal{L}(\phi_{ES}, \phi_{ER}, 10^{-3}s) + t_a < \rho$, the wireless network delay between the front (ϕ_{ES}) and the rear (ϕ_{ER}) vehicle plus actuation time (t_a) should be less than ρ. Otherwise, the emergency braking system acts late and the vehicle collides with the one in front.

7.3.1 Power Grid Synchronization (Synchrophasor)

In order to reconnect a generator to the power grid for distribution of Alternating Current (AC) power, the generator should be synchronized to the system parameters to ensure voltage and frequency stability. When power components providers are connecting to the AC grid, their voltage (amplitude), frequency and phase must match. As the time related specification of PMU (Power Management Unit), the frequency of two sources should be 50 or 60 Hz with maximum tolerance of 1% and the maximum allowed phase deviation is about 10 degrees. Since frequency and phase parameters are time sensitive, it is possible to specify timing constraints on them.

Modeling of Synchrophasor Using Two DC Motors

Power grid synchronization has been modeled using two DC motors where the first motor (master motor) represents the grid reference for frequency and phase, and the second one (slave motor) demonstrates the generator which should be controlled by the master motor. Figure 7.5 shows the schematic of controlling two DC motors. Two motors are controlled by two Arduino Mega 2560 boards synchronized by wireless modules (NRF24L01). The phase of motors are monitored by two cRIO platforms (figure 7.5).

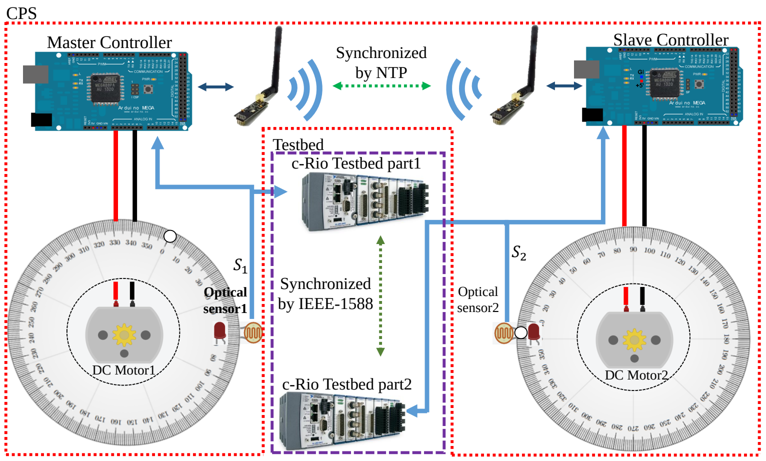


Figure 7.5: The Schematic of Controlling Two DC Motors to Rotate in the Same Phase.

Two dials labeled from 0 to 360 degrees are installed on the motors' shafts to illustrate two sinusoidal signals. The angular speed of reference motor is set to 60 revolutions per second which indicates the power grid frequency (60 Hz). A small hole is drilled on both dials at zero degrees in order to detect a revolution using photomicrosensor. The goal is to synchronize the speed (frequency) and phase of the slave motor with the master after an activation event rises. The setup for two DC motors is depicted in figure 7.6. In this setup, an Arduino Mega 2560 board is used for each motor, and they are implemented as a distributed synchronization system.

Two Arduino boards are connected by two wireless modules (NRF24L01+, 2.4 GHz) by which the master motor controller sends required data to the slave controller. Moreover, the master controller has an additional role for the slave and the slave uses it as the reference for clock synchronization by NTP [Mills, 1989]. Using NTP, the two devices can be synchronized to a precision of about 2 ms through the exchange of NTP messages.

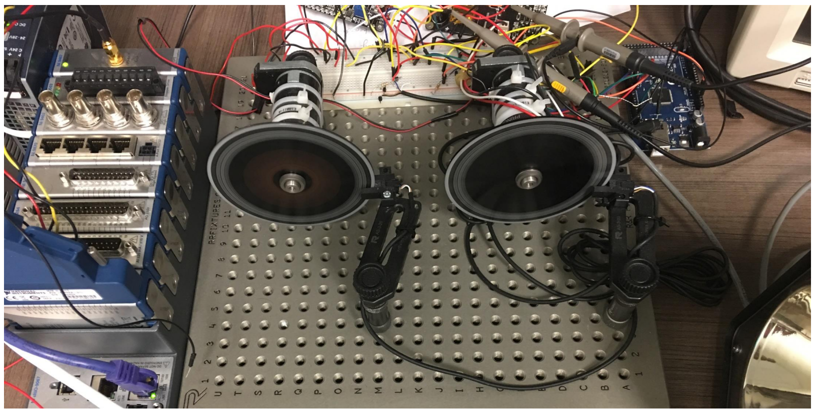


Figure 7.6: Modeling the Synchrophasor Using Two DC Mtors.

Since the CPS implementation is in a distributed manner, the testbed should be distributed as well (Testbed part 1 and 2 in figure 7.5). One cRIO controller is dedicated for each motor to monitor the sensors' signals.

In a distributed system, having a common understanding of time is a critical feature of the testbed. In order to achieve this capability, cRIO controllers use the NI-TimeSync plug-in that utilizes the IEEE 1588 Precision Time Protocol (PTP)[IEEE Instrumentation and Measurement Society, 2002] as the synchronization protocol with 100 *ns* precision utilizing a Local Area Network (LAN).

An NI-9381 module is used on each cRIO 9067 which contains 37 pins including eight analog input/output and four digital input/output pins. Two analog input pins on NI-9381 are used and connected to the sensor output pins on each motor. Two installed sensors, Omron EESX970C1, have 5 V as their output when they detect the hole. Once the sensor output crosses 2.5 V from below, a hole is detected (the threshold is 2.5 V).

Timing Specifications of Synchrophasor

In such a synchrophasor, a pair of generators connected to the same grid should generate a sinusoidal signal with frequency, $f = 60$ Hz, with 0.24% tolerance[Fre]. The phase difference between two generators should not be greater than $10° \pm 0.5°$. The timing constraint for this case study is a frequency constraint on both sinusoidal signals and a phase constraint between them. One can write the frequency constraint as $\mathcal{F}(\phi_{Master}, 0.144) = 60$ where acceptable tolerance for period, T, is $\frac{1}{60+0.144} \leq T \leq \frac{1}{60-0.144}$ ($16.63\ ms \leq T \leq 16.67$ ms) so the tolerance error window is about 80 μs. For the second requirement, the phase between the signals should be within 10°. So, the phase constraint can be described by $\mathcal{P}(\phi_{Master}, \phi_{Slave}, 23\ \mu s) < 0.463\ \mu$s. Acceptable tolerance for this requirement is $\varepsilon = 463\ \mu$s because a complete revolution (360°) is done in 16.67 ms ($\frac{1}{60\ Hz} = 16.67\ ms$).

In order to test the timing of this application, its timing specifications should be defined clearly. I defined the timing constraints for this case study as *"The frequency of the rising edges of two signals s_1 and s_2 from the master and the slave sensors crossing threshold, 2.5 V, must be 60 Hz and the time at which two sensors detect the drilled hole should be exactly the same in each period with at most 463 μs error. The tolerances are 0.144 Hz and 23μs for the frequency and phase respectively"*.

The TTL statement for master controller is (s_1 and s_2 are the signals on *Master* and *Slave* respectively):

$$\mathcal{F}(\langle s_1, 2.5, \nearrow\rangle, 0.144Hz) = 60\ Hz$$

Similarly, we can write the TTL statement for the slave controller as:

$$\mathcal{F}(\langle s_2, 2.5, \nearrow\rangle, 0.144Hz) = 60\ Hz$$

and as a distributed system we have:

$$\mathcal{P}(\langle s_1, 2.5, \nearrow\rangle, \langle s_2, 2.5, \nearrow\rangle, 0.000023s) < 0.0004\ s$$

Three aforementioned TTL expressions can be rewritten by a single statement separated with logic AND($\wedge$). The monitoring program evaluates results online.

7.3.2 Simultaneous Image Capturing for 3D Reconstruction

3D image reconstruction based on multiple 2D images taken from different angles of a scene has application in many fields, including entertainment, military (geometric information extraction), autonomous driving, and sports (e.g., football match analysis). 100 μs is taken as the maximum delay between capture time of two cameras. This requirement is interpreted as a *Simultaneity* constraint between trigger signals of cameras with maximum acceptable tolerance of $\varepsilon = 10\ ms$ or $\mathcal{S}(\phi_{c_1}, \phi_{c_2}, 10\ ms)$.

Modeling of Simultaneous Image Capturing Application

A model of the application was implemented where each camera takes a picture of a rolling ball from different angles at the same time. Figure 7.7 depicts the experiment testbed as well as the monitoring platform. The equipment are ArduCAM ESP8266 UNO boards which include a 2 MP CMOS camera for image capturing, a built-in ESP8266 module for wireless communication and some digital I/O. The connected battery was a rechargeable 3.7 V 700 mAh Li-Po battery as the power supply. A web-server is used to send the capture command to both cameras. Upon capturing, each ArduCAM board generates a trigger signal on one of the digital input/output

(I/O) pins. This signal is used to show the delay between capture time instances between two cameras. Figure 7.7 shows the ArduCAM boards (towards the bottom) taking pictures of a rolling soccer ball and the testing platforms.

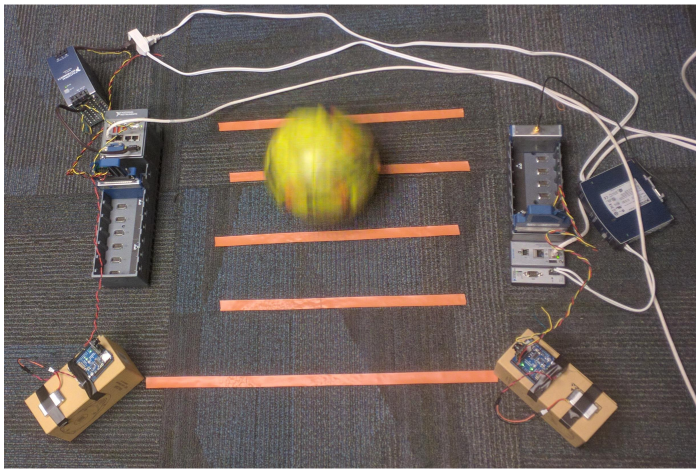
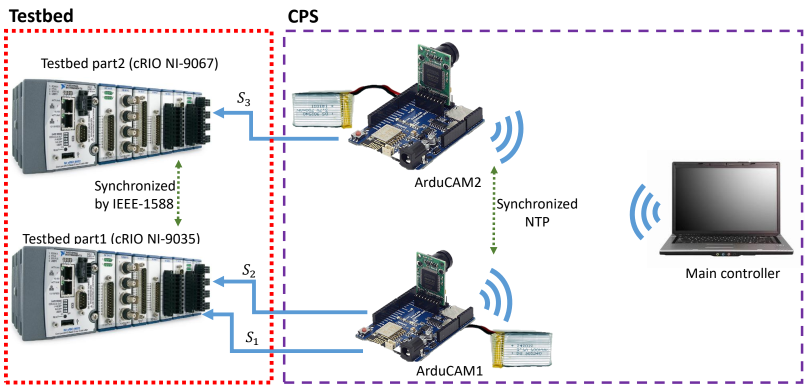


Figure 7.7: Two ArduCAM Taking Simultaneous Images of a Moving Soccer Ball.

As Figure 7.8 demonstrates, there is a server (a desktop computer or a laptop) with a wireless communication device to send the command to camera boards. Once each camera receives the message from the server, they should take a picture after no more than 0.2 s and send it back to the server. Then, these images can be used for 3D image reconstruction. In order to avoid blurring, all images should be taken simultaneously. ArduCAM1 (the bottom camera in figure 7.8) raises a flag on its digital pin #2 once it receives the command. This pin is connected to the testbed as signal s_1. Each ArduCAM board is set to raise a flag on one of digital pin #3 when they take the photo. The pin #3 of each camera is connected to the corresponding cRIO device as signals s_2 and s_3. Applying the configuration from the first case study, each camera is attached to one of the cRIO devices. Since we defined that the cameras must take the pictures synchronously, the events detected on s_2 and s_3 should occur simultaneously within 0.2 s after they receive the command message. Thus, the time interval between the event detected on signals s_1 and s_2 should be less than 0.2 s.


Testbed
CPS
Testbed part2 (cRIO NI-9067)
S_3
Synchronized by IEEE-1588
Testbed part1 (cRIO NI-9035)
S_2
S_1
ArduCAM2
Synchronized NTP
Main controller
ArduCAM1
Figure 7.8: The Schematic of Monitoring Simultaneous Image Capturing.

7.3.3 Timing Specifications of Simultaneous Image Capturing

In specifying the timing constraints, the requirement can be specified as a maximum *Latency* and a *Simultaneity* constraints within a time error tolerance. Hence, the timing specification is, *The signal s_2 should go above* $2.5\ V$ *at the same time with signal s_3 when it goes above* $2.5\ V$ *with* $0.01\ s$ *tolerance* and *the latency between the time that the activation signal, s_1, goes above* $2.5\ V$ *and the time that the capture signal, s_2, goes above* $2.5\ V$ *should be less than* $0.2\ s$ *with tolerance of* 0.01.

In TTL:

$$(\mathcal{S}(\langle s_2, 2.5v, \nearrow\rangle, \langle s_3, 2.5v, \nearrow\rangle, 0.01s) \wedge (\mathcal{L}(\langle s_1, 2.5v, \nearrow\rangle, \langle s_2, 2.5v, \nearrow\rangle, 0.01s) < 0.2s))$$

7.4 Applying TTL Reasoning System on Flying Paster and RIM

In this section, the benefits of TTL reasoning system for monitoring of temporal behavior of CPS are demonstrated.

7.4.1 Consistency Checking of Temporal Specifications

TTL logic can check the consistency between timing specification that are expressed for a system.

The following specifications for flying paster are considered.

From physics rules, it is known that the relation between linear and angular velocity as $\omega_S = \frac{v_S}{r_S}$ and between the angular velocity and the frequency of rotation: $\omega_S = 2\pi F_S \Rightarrow v_S = 2\pi r_S F_S$. Besides, from the system definition, we know that the linear velocity of both rolls should equal, $v_S = v_A$. Hence:
$2\pi r_S F_S = v_A \Rightarrow F_S = \frac{v_A}{2\pi r_S} Hz \pm 0.5\%$ (F_S is frequency of S), $r_S = 0.045m$ and the acceptable error is 0.5% of F_S.

$$\mathcal{F}\left(\phi_\gamma, 0.005 \times \frac{v_A}{2\pi r_S}\right) = \frac{v_A}{2\pi r_S}$$

Also, from the other definitions, we know:

$tapeToContactAngle = \frac{225}{360} \times \lambda$ and $tapeToCutAngle = \frac{270}{360} \times \lambda$

λ is the time of one rotation of S. It is necessary to check if the latency between Contact and Cut is greater than 1.5 ms or in TTL $\mathcal{L}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s) < 1.5ms$. The TTL reasoning to check the consistency between those four specifications is below:

1	$\mathcal{F}(\phi_{\gamma}, 0.005 \times \frac{v_A}{2\pi r_S}) = \frac{v_A}{2\pi r_S}$	
2	$\tau(\phi_{Contact}) = \frac{225}{360} \times \frac{\lambda}{\delta}$	
3	$\tau(\phi_{Cut}) = \frac{270}{360} \times \frac{\lambda}{\delta}$	
4	$\mathcal{L}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s) < 1.5ms$	
5	$\tau(\phi_{\gamma}^{n}) - \tau(\phi_{\gamma}^{n-1}) \leq \frac{2\pi r_S}{0.995 v_A \delta} - 1$	FE, 1
6	$\frac{2\pi r_S}{1.005 v_A \delta} + 1 \leq \tau(\phi_{\gamma}^{n}) - \tau(\phi_{\gamma}^{n-1})$	FE, 1
7	$\tau(\phi_{\gamma}^{n}) - \tau(\phi_{\gamma}^{n-1}) = \frac{360}{225} \times \tau(\phi_{Contact})$	Q1, 2
8	$\tau(\phi_{\gamma}^{n}) - \tau(\phi_{\gamma}^{n-1}) = \frac{360}{270} \times \tau(\phi_{Cut})$	Q1, 3
9	$\frac{360}{225} \times \tau(\phi_{Contact}) \leq \frac{2\pi r_S}{0.995 v_A \delta} - 1$	A7, 5, 7
10	$\frac{2\pi r_S}{1.005 v_A \delta} + 1 \leq \frac{360}{270} \times \tau(\phi_{Cut})$	A7, 6, 8
11	$\frac{360}{225} \times \tau(\phi_{Contact}) < \frac{2\pi r_S}{0.995 v_A \delta}$	A10, 9
12	$\frac{2\pi r_S}{1.005 v_A \delta} < \frac{360}{270} \times \tau(\phi_{Cut})$	A10, 10
13	$-\frac{2\pi r_S}{0.995 v_A \delta} < -\frac{360}{225} \times \tau(\phi_{Contact})$	A5, 11
14	$\frac{270}{360} \times \frac{2\pi r_S}{1.005 v_A \delta} < \tau(\phi_{Cut})$	Q3, 12
15	$-\frac{225}{360} \times \frac{2\pi r_S}{0.995 v_A \delta} < -\tau(\phi_{Contact})$	Q3, 13
16	$\frac{2\pi r_S}{360 v_A \delta}(\frac{270}{1.005} - \frac{225}{0.995}) < \tau(\phi_{Cut}) - \tau(\phi_{Contact})$	A4, 14, 15
17	$1.68ms < \mathcal{L}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s)$	LMI, 16

Line 17 shows that the latency between **Contact** and **Cut** is greater than $1.68ms$ which is **inconsistent** with $\mathcal{L}(\phi_{Contact}, \phi_{Cut}, 10^{-4}) < 1.5ms$ (the third temporal specification in Section 7.5.1).

7.4.2 Consistency Checking for Autonomous Intersection Scenario

In the proposed scenario in figure 7.4, when two CAVs are driving on the same lane and same direction, the deceleration on front vehicle is detected in the rear by either receiving the sudden brake message (ϕ_{ER}) sent through the wireless network or measuring distance by sensor which sends ϕ_{SS} signal as an event and receive ϕ_{SR} event by reflection 7.4.b. If rear vehicle detects an inconsistency between the measured velocity (calculated using the sensors' data) and the velocity received through DSRC, it should drive more conservative (maintain a larger distance) to avoid any accidents (larger d_{min}). Another option is reducing the response time of the rear vehicle. The first solution degrades the throughput of the highway while the second one needs re-scheduling of corresponding tasks in the ECUs and increase the sampling rate of sensors if possible. By using a reasoning system, it is possible to derive a relation between parameters of system and prove the satisfaction or violation of a safety requirement from the available system information (equation 7.1). Moreover, it allows for developing fault-tolerant routines when a requirement is not going to be met.

Regarding the second option, in the case of increasing v_f or having a mismatch between the received and the calculated v_f in rear vehicle (by sensors e.g. Radar), the rear car should be more conservative. As a fact, the actual end-to-end delay for reading the sensor (detecting v_f) should be less than ρ: $(t_{SR}-t_{SS})+WCET_S+t_a < \rho$. t_{SR} and t_{SS} are the time at which sensor sends and receives a signal to measure the distance. Figure 7.4.c depicts the equation by a block diagram which is the block diagram of decision making on the rear vehicle to know when a deceleration should be applied. $WCET_S$ is the worst case execution time of calculation to find v_f and t_a is the brake actuation time. Furthermore, another message, *sudden brake* (ϕ_{EB} event), is issued as a network message by the front vehicle in case of braking (figure 7.4.b). Once the rear vehicle receives the message, it decelerates. We have $t_{EBR}-t_{EBS}+t_a < \rho$ where t_{ER} and t_{ES} are the time of receiving and sending the sudden brake message from the front vehicle.

A)

$$\tau(\phi_{SR}) - \tau(\phi_{SS}) + \frac{WCET_S + t_a}{\delta} < \frac{\rho}{\delta}$$

B)

$$\tau(\phi_{ER}) - \tau(\phi_{ES}) + \frac{t_a}{\delta} < \frac{\rho}{\delta}$$

1	$\tau(\phi_{SR}) - \tau(\phi_{SS}) + \frac{WCET_S+t_a}{\delta} < \frac{\rho}{\delta}$	
2	$\tau(\phi_{ER}) - \tau(\phi_{ES}) + \frac{t_a}{\delta} < \frac{\rho}{\delta}$	
3	$\tau(\phi_{SR}) - \tau(\phi_{SS}) < \frac{\rho-(WCET_S+t_a)}{\delta}$	A9, 1
4	$\tau(\phi_{ER}) - \tau(\phi_{ES}) < \frac{(\rho-\varepsilon_l)+\varepsilon_l-t_a}{\delta}$	A9, 2
5	$\frac{1}{\rho-(WCET_S+t_a)+5} + 6 < \mathcal{F}(\phi_S, 5)$	FXI, 3
6	$\mathcal{L}(\phi_{ES}, \phi_{ER}, 10^{-3}s) < \rho - 10^{-3}s - t_a + 1$	LXI, 4
7	$\frac{1}{\rho-(WCET_S+t_a+5} + 6 < \mathcal{F}(\phi_S, 5) \wedge \mathcal{L}(\phi_{ES}, \phi_{ER}, 10^{-3}s) < \rho - t_a + 0.999$	$\wedge$I, 7, 8

The proposition in line 3 is translated into frequency (line 5) since the distance is periodically measured and communicated by the sensor (ϕ_{SS}). Within the specified transmission latency, the measurements are received and the response formulated (ϕ_{SR}) in consecutive events. Line 5 determines the frequency of sending sensor measurement signals proportional to ρ. Hence, decreasing the response time becomes possible, by increasing the rate of sensor measurement (e.g. Radar). Line 6 denotes another boundary for decreasing ρ. Thus, if the monitored average delay of wireless network is more than a certain value, the brake within response time is not going to happen in the near future so that we conclude the vehicle is driving unsafely now. By the last statement (line 7), the deduction elaborates the **consistency** between the frequency of reading sensor and the wireless network delay for emergency braking.

7.4.3 Considering Measurement Errors for Accurate Monitoring

Dealing with uncertainties during the operation of CPS is required and it should be considered in both designing and verification stages. That is because they exist in all parts of CPS including physical units, network infrastructure or even processing. If they are not taken into account, the CPS operation does not follow the specifications defined in its design. On the other side, the verifying process should be aware of existing measurement errors to have the right verification for the system operation. In this chapter, the effects of considering uncertainties in CPS operation are scrutinized and the result are compared with the real experiments [1]

For the flying paster case study presented in section 7.1.1, it is found that there is an inconsistency for the safety requirement $\mathcal{L}(\phi_{Contact}, \phi_{cut}, 10^{-4}s) < 1.5ms$ and the reasoning showed that $\mathcal{L}(\phi_{Contact}, \phi_{cut}, 10^{-4}s)$ should be greater than $1.5141ms$. It means that it should inevitably changed to a valid specification. Here, it is changed to

$$3ms < \mathcal{L}(\phi_{Contact}, \phi_{cut}, 10^{-4}s)$$

(if it is greater than 3 ms it is definitely greater than 1.5) to make it consistent with the rest timing constraints and then see the effect of considering user defined tolerance and measurement error in the calculation.

Now, let's look at the proof in section 7.4.1 again. Line 1 contains the uncertainty in frequency. Accordingly, its effect on line 16 is: $\frac{2\pi r_S}{360 v_A \delta} \times 42.52 < \tau(\phi_{Cut}) - \tau(\phi_{Contact})$ since the latency between **Contact** and **Cut** should be greater than 3ms, it is enough the left part of the above equation is greater than $3ms$: $0.003 < \frac{2\pi r_S}{360 v_A} \times 42.52$. Considering $\pi = 3.14$ and $r_S = 4.5cm$, we have, $v_A < 11.12m/s$. If the tolerance and measurement error are not considered, the tolerance is eliminated in frequency (line 1) and both 1.005 and 0.995 are replaced by 1 in line 16. Since we do not consider the tolerance and measurement error in latency as well, the

[1] In order to see the effects of error measurement, it should be big enough to be seen. Therefore, in the implementation of case studies it has been decreased to $200\mu s$.

equation is: $\frac{2\pi r_S}{360 v_A} \times 45 < \tau(\phi_{Cut}) - \tau(\phi_{Contact})$. Hence, $0.003 < \frac{0.785 r_S}{v_A}$.

In case of removing the tolerance and error the threshold for v_A is $v_A < 11.775 m/s$. An analog to digital input/output data acquisition module monitors the time delay between *Contact* and *Cut* for ten different linear speeds for A. The collected data depicted in figure 7.9 confirms that in speeds greater than $11.12 m/s$, the delay between Contact and Cut is less than $3ms$. The purple line in the figure shows that there is w m/s error when the measurement error is considered. If δ in the reasoning process is not taken into account, for all velocities between the pink line and brown line, the latency between **Contact** and **Cut** is not met. However, the calculation without δ shows they are met.

Furthermore, based on the reported data by the data acquisition system, the actual latency between **Contact** and **Cut** , the calculated value with and without considering tolerances in table 7.1. The table is about the comparison between the calculated delay using deduction and measured delay (in experiments) between **Contact** and **Cut** events (in microseconds). In each column (scenario), the numbers for **Measured** and **Calculated with Tolerance** are closer than **Measured** and **Calculated without Tolerance**. This demonstrates that not considering the tolerance can cause a problem because there is a gap between the calculation and real implementation.

Apparently, the measured data is closer to the calculation involving the measurement error.

Table 7.1: Seven Scenarios in Flying Paster to See the Effect of Measurement Error in the Calculation.

	8 m/s	**9 m/s**	**10 m/s**	**11 m/s**	**12 m/s**	**13 m/s**	**14 m/s**
Calculated with Tolerance in μs	4172.275	3708.689	3337.82	3034.382	2781.517	2256.554	2384.157
Calculated without Tolerance in μs	4415.625	3925	3532.5	3211.364	2943.75	2717.308	2523.214
Measured in μs	4240	3736	3400	2912	2672	2544	2352

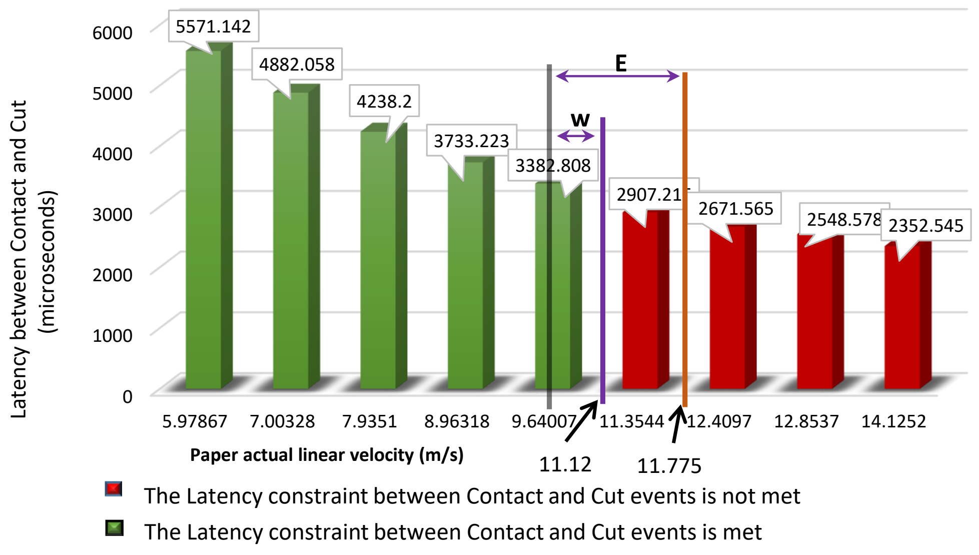


Figure 7.9: Latency Measurement between Contact and Cut When the Velocity of Active Roll Is Increased. Latency Measurement between Contact and Cut. w Is the Error Value between Actual Velocity and the Calculated by the Reasoning System with Considering Measurement Errors. E Demonstrates the Error Value When We Do not Consider the Measurement Errors. Ignoring the Measurement Error Can Result in an Incorrect Evaluation of the Timing Constraint and Cause a Failure.

7.4.4 Simplifying the Monitoring Logic

One solution to verify CPS specifications is through monitoring the temporal behavior at runtime. The simplicity of timing specfciations can reduce the effort required to design the monitoring logic. Additionally, if we can decrease the number of specifications and/or their complexity, the memory and power consumption of the monitoring logic implementation can be diminished. Since most monitoring devices are implemented on FPGAs and they have relatively smaller memory, simplification of specification is not an option it is a necessity. In order to illustrate this, there are two examples in flying paster and RIM.

Flying Paster Application

To reduce the complexity of the monitoring logic for the flying paster case study, there is an example here:

1	$\mathcal{C}(\phi_{Match}, \phi_{TDC}, 10^{-4}s)$	
2	$\tau(\phi_{\text{TDC}}) - \tau(\phi_{\text{Match}}) < \frac{2\pi}{\omega\delta}$	
3	$\tau(\phi_{\text{TDC}}) + \frac{4\pi}{\omega\delta} + \frac{270\pi}{180\omega\delta} < \tau(\phi_{\text{Cut}})$	
4	$\frac{4\pi}{\omega\delta} + \frac{225\pi}{180\omega\delta} < \tau(\phi_{Contact}) - \tau(\phi_{Match})$	
5	$\tau(\phi_{Contact}) - \tau(\phi_{TDC}) < \frac{4\pi}{\omega\delta} + \frac{225\pi}{180\omega\delta}$	
6	$\tau(\phi_{Cut}) - \tau(\phi_{Match}) < \frac{6\pi}{\omega\delta} + \frac{270\pi}{180\omega\delta}$	
7	$\tau(\phi_{Cut}) + \frac{4\pi}{\omega\delta} + \frac{225\pi}{180\omega\delta} < \tau(\phi_{Contact}) + \frac{6\pi}{\omega\delta} + \frac{270\pi}{180\omega\delta}$	A4, 4, 6
8	$\tau(\phi_{\text{Cut}}) - \tau(\phi_{\text{Contact}}) < \frac{2\pi}{\omega\delta} + \frac{45\pi}{180\omega\delta}$	A9, 7
9	$\tau(\phi_{Contact}) + \frac{4\pi}{\omega\delta} + \frac{270\pi}{\omega\delta} < \tau(\phi_{Cut}) + \frac{4\pi}{\omega\delta} + \frac{225\pi}{180\omega\delta}$	A4, 3, 5
10	$\frac{45\pi}{180\omega\delta} < \tau(\phi_{Cut}) - \tau(\phi_{Contact})$	A9, 9
11	$14 < \tau(\phi_{Cut}) - \tau(\phi_{Contact})$	A8, 10
12	$\mathcal{C}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s)$	CHI, 11
13	$\mathcal{L}(\phi_{Contact}, \phi_{Cut}, 10^{-4}s) < \frac{2\pi}{\omega} + \frac{45\pi}{180\omega}$	LXI, 8

Here, there are six hypotheses as system specifications. At the end, there are two deduced independent timing specifications on just **Contact** and **Cut** events (line 12 and 13). If line 13 is true since **Cut** should occur after **Contact** with the tolerance of $\varepsilon = 10^{-4}s$, line 12 is also true. Thus, instead of monitoring 6 timing specifications on 4 events, the monitoring can be done just on one timing specification on 2 events (**Contact** and **Cut**) plus first two specifications (lines 1 and 2 that are not used in the proof). Therefore, the number of timing specifications to monitor is reduced in half and therefore, the required area for implementation is accordingly deducted.

Autonomous Intersection Manager

In RIM, when a vehicle receives the assigned VoA/ToA, it is responsible to satisfy its specified constraints using the vehicle's controller. Basically, when IM receives a request, calculates VoA/ToA for the new vehicle based on the other vehicles' trajectories meaning if IM already assigned VoA/ToA to n vehicles, it should calculate them for $vehicle_{n+1}$ as well. And then, it checks the calculated trajectory with those of $vehicle_1$ to $vehicle_n$. If there is a single conflict or more, IM should figure out another safe pair of VoA/ToA for $vehicle_{n+1}$.

On the other side, when $vehicle_{n+1}$ receives the assigned VoA/ToA (at time t_θ in figure 7.2) there should be an enough time to achieve them through its controller otherwise, it increases the chance of accident inside the intersection area. Therefore, $t_{IL} - t_\theta$ (in figure 7.2) should be greater than a threshold. Based on these information and TTL reasoning, we can determine the specifications of the vehicle, and monitor the vehicle behavior to verify the system correctness. As RIM specifies, the Request and Intersection Lines (RL and IL) are fixed and their distance is given as Len (figure 7.2). Also, the minimum and maximum velocity of road connected to the intersection are known, i.e. V_{min} and V_{max}. Therefore, we are aware of the latest and earliest time by which a vehicle can travel this distance: $\frac{len}{V_{\max}} < t_{IL} - t_{RL} < \frac{len}{V_{\min}}$

Moreover, we assume network delays between vehicle and intersection are bounded:

$t_{receive_{IM}} - t_{req_{vehicle}} < max_s$ and $t_\theta - t_{send_{IM}} < max_r$

Hence, they can be expressed by a single variable max_N, $max_N = max_r + max_s$. we know $th = \frac{V_{\max} - V_{\min}}{a_{\max}}$, and the time between receiving VoA/ToA (at t_θ) and the time to reach IL can be find by a subtraction on the time of passing RL to IL and the time between RL and t_θ.

1	$th < \mathcal{L}(\theta, IL, 10^{-3}s)$	
2	$\mathcal{L}(RL, IL, 10^{-3}s) < \frac{len}{V_{\min}}$	
3	$\frac{len}{V_{\max}} < \mathcal{L}(RL, IL, 10^{-3}s)$	
4	$\mathcal{L}(\theta, IL, 10^{-3}s) = \mathcal{L}(RL, IL, 10^{-3}s) - \mathcal{L}(RL, \theta, 10^{-3}s)$	
5	$\mathcal{L}(RL, \theta, 10^{-3}s) = \mathcal{L}(receive_{IM}, send_{IM}, 10^{-3}s) + \max_N$	
6	$th + \frac{len}{V_{\max}} < \mathcal{L}(\theta, IL, 10^{-3}s) + \mathcal{L}(RL, IL, 10^{-3}s)$	A4, 1, 3
7	$th + \frac{len}{V_{\max}} - \mathcal{L}(\theta, IL, 10^{-3}s) < \mathcal{L}(RL, IL, 10^{-3}s)$	A9, 6
8	$th + \frac{len}{V_{\max}} - \mathcal{L}(\theta, IL, 10^{-3}s) < \frac{len}{V_{\min}}$	A10, 7, 2
9	$th + \frac{len}{V_{\max}} < \frac{len}{V_{\min}} + \mathcal{L}(\theta, IL, 10^{-3}s)$	A9, 8
10	$th + \frac{len}{V_{\max}} - \frac{len}{V_{\min}} < \mathcal{L}(\theta, IL, 10^{-3}s)$	A9, 9
11	$th + \frac{len}{V_{\max}} - \frac{len}{V_{\min}} < \mathcal{L}(RL, IL, 10^{-3}s) - \mathcal{L}(RL, \theta, 10^{-3}s)$	A10, 4, 10
12	$th + \frac{len}{V_{\max}} - \frac{len}{V_{\min}} + \mathcal{L}(RL, \theta, 10^{-3}s) < \mathcal{L}(RL, IL, 10^{-3}s)$	A9, 11
13	$th + \frac{len}{V_{\max}} - \frac{len}{V_{\min}} + \mathcal{L}(RL, \theta, 10^{-3}s) < \frac{len}{V_{\min}}$	A10, 12, 2
14	$\mathcal{L}(RL, \theta, 10^{-3}s) < \frac{len}{V_{\min}} - (th + \frac{len}{V_{\max}} - \frac{len}{V_{\min}})$	A9, 13
15	$\mathcal{L}(receive_{IM}, send_{IM}, 10^{-3}s) + max_N < \frac{2len}{V_{\min}} - (th + \frac{len}{V_{\max}})$	A7, 14, 5
16	$\mathcal{L}(receive_{IM}, send_{IM}, 10^{-3}s) < \frac{2len}{V_{\min}} - (th + \frac{len}{V_{\max}} + \max_N)$	A9, 15

The above deduction illustrates that the process for responding to a request must be bounded by a threshold $(\frac{2len}{V_{\min}} - (th + \frac{len}{V_{\max}} + \max_N) + \varepsilon)$. Otherwise, the intersection safety is not guaranteed. The IM checks for conflicts in trajectory of the requesting vehicle and the existing vehicles inside the intersection area. As the number of existing vehicles increases, the IM processing time increases exponentially. Hence, the desired safety specification is violated. As the proof shows, it is possible to monitor just one timing specification (line 16) on two events instead of monitoring 5 specifications on 5 different events to achieve the same goal.

To compare the required space for implementing the simplified timing specifica-

Table 7.2: The Number of Required Lookup Tables and Flip-flops for Implementing Monitoring Tool on Zynq 7000.

	flying paster Timing Specifications			Autonomous Intersection Timing Specifications		
	All	Simplified	Memory Reduction	All	Simplified	Memory Reduction
#LUTs	1450	889	38.6%	1263	632	49.9%
#FFs	2250	1320	41.3%	1940	910	53%

tions with initial ones, the monitoring logic has been implemented using the TMA [2] . Table 7.2 shows the required memory for implementing all specifications versus the simplified temporal specifications on Xilinx Zynq 7000. Thus, by monitoring simplified temporal specifications, we can reduce the memory requirement for implementing on an FPGA by about 45.7%.

7.4.5 Reasoning about Unobservable Events

Not all events inside a CPS can be monitored, but their timing may be needed to analyze the overall system behavior. For example, in the flying paster, after the AoP event, the linear velocity of S should be the same as A. When this condition is satisfied, $Match$ is issued. If the time difference between AoP and Match is too long, it shows that the settling time of the controller for the spare roller (S) has exceeded the latency constraint. As a result, the active roll may run out of paper before the replacement is made. If we can monitor the time for AoP and determine the time for Match, the monitoring system can predict the settling time for the controller. As it is mentioned earlier, $t_{termination} = 6s$.

[2]Explained in Chapter 5.

1	$\mathcal{C}(\phi_{TDC}, \phi_{Match}, 10^{-4}s)$	
2	$\mathcal{L}(\phi_{\text{AoP}}, \phi_{\text{Cut}}, 10^{-4}s) < 6s$	
3	$\frac{10^{-4}s}{\delta} - 1 < \tau(\phi_{\text{TDC}}) - \tau(\phi_{\text{Match}})$	CHE, 1
4	$\tau(\phi_{\text{Cut}}) - \tau(\phi_{\text{AoP}}) < \frac{6s-10^{-4}s}{\delta} + 1$	LXE, 2
5	$\tau(\phi_{Cut}) - \tau(\phi_{AoP}) < \tau(\phi_{TDC}) - \tau(\phi_{Match}) + \frac{6s-2\times10^{-4}}{\delta} + 2$	A4, 3, 4
6	$\tau(\phi_{Match}) - \tau(\phi_{AoP}) < \tau(\phi_{TDC}) - \tau(\phi_{Cut}) + \frac{6s-2\times10^{-4}}{\delta} + 2$	A9, 5
7	$\mathcal{L}(\phi_{AoP}, \phi_{Match}, 10^{-4}) < \delta\tau(\phi_{TDC}) - \delta\tau(\phi_{Cut}) + 6s - 10^{-4}s + \delta$	LXI, 6

As the example is showing, the deduction system can find a maximum band for the latency between **AoP** and **Match**.

7.5 Using TMA for Run-time Monitoring of Simultaneous Image Capturing and Synchrophasor Applications

7.5.1 Monitoring Flying Paster Using TMA

Based on the desired operation of the flying paster, its timing specifications are converted to STL as follows. Noted that:

- A: Active roll.
- S: Spare roll.
- v: linear velocity.
- r: radius.
- ω: angular velocity.
- t_{action}: the duration from t_{AoP} to t_{Match}.

- $t_{termination}$: the duration between t_{AoP} to t_{Cut}.

1) The velocity of the paper on active roll should be constant:

$$v_A = (r_A \times \omega_A) \pm 1\% \ \ \text{m/s}$$

$$\square_{[t_i, t_s]}(v_A = r_A \times \omega_A \pm 1\%)$$

2) The time interval between AOP rising to $Match$ rising edge must be no more than t_{action}: $\square(\uparrow AOP \Rightarrow \lozenge_{[0,t_{action}]}(\uparrow Match))$

3) After $match$, the paper speed of the spare should remain the same as active: $v_A = r_A \times \omega_A$ and $v_S = v_A \pm 1\%$

$$\square_{[t_{Match}, t_{Cut}]}(v_A = r_A \times \omega_A)$$
$$\square_{[t_{Match}, t_{Cut}]}(v_S = r_S \times \omega_S)$$
$$\square_{[t_{Match}, t_{Cut}]}(v_S = v_A \pm 1\%)$$

4) Catch the TDC (2 rotations of A after $Match$).

$$t_{TDC} - t_{Match} < \tfrac{4\pi}{\omega_S}$$
$$\lozenge_{[t_{Match}, t_{Match}+\frac{4\pi}{\omega_S}]}(\uparrow TDC)$$

5) When tape is 225 degrees after TDC, *Contact* signal must fire.

$$t_{Contact} - (t_{TDC} + \frac{225\ degrees}{\omega_S}) < \pm 1\ ms.$$
$$\square_{[t_{TDC}+\frac{225\ degrees}{\omega_S+1\ ms}, t_{TDC}+\frac{225\ degrees}{\omega_S-1\ ms}]}(\uparrow Contact)$$

6) When tape is 270 degrees after TDC, *Cut* signal must fire.

$$t_{cut} - (t_{spareTDC} + \frac{270\ degrees}{\omega_S}) < \pm 1\ ms$$
$$\square_{[t_{spareTDC}+\frac{270\ degrees}{\omega_S \pm 1\ ms}, t_{spareTDC}+\frac{290\ degrees}{\omega_S \pm 1\ ms}]}(\uparrow cut)$$

7) AOP to Cut should not be more than $t_{termination}$ (The user defines $t_{termination}$ and it is the maximum duration in which the roll changing should be done. In this scenario it is $6s$).

$$\Diamond_{[t_{AOP},t_{AOP}+t_{termination}]}(\uparrow Cut)$$

For the evaluation, the timing constraints of Flying Paster with three approaches (conventional, Jakšić [Jakšić*et al.*, 2015] and TMA) have been implemented on cRIO FPGA board. In the conventional technique, the horizon for each operator is stored in memory and the signal history is used for the evaluation. For synthesising the existing method [Jakšić*et al.*, 2015], all future STL formulas have been converted to the past notation by the method explained in its original paper. Finally, the same timing constraints in TTL synthesized in TMA. For example, in $\Box(\uparrow AOP \Rightarrow \Diamond_{[0,t_{action}]}(\uparrow Match))$, we have:

Jakšić and conventional Method (which is pointed out as *Register Buffer* in [Jakšić*et al.*, 2015]) use STL: Since rising and falling edges ($\uparrow$ and $\downarrow$) cannot be represented in STL, we express them as the way in [Maler and other, 2013]:

$$\uparrow \psi = (\psi \wedge (\neg\psi \ \mathcal{S}\ T)) \vee (\neg\psi \wedge (\psi \ \mathcal{U}\ T))$$
$$\downarrow \psi = (\neg\psi \wedge (\psi \ \mathcal{S}\ T)) \vee (\psi \wedge (\neg\psi \ \mathcal{U}\ T))$$

Therefore, the example is converted to:

$$\Box((AOP \wedge (\neg AOP \ \mathcal{S}\ T)) \vee (\neg AOP \wedge (AOP \ \mathcal{U}\ T)) \Rightarrow$$
$$\Diamond_{[0,t_{action}]}(Match \wedge (\neg Match \ \mathcal{S}\ T) \vee (\neg Match \wedge (Match \ \mathcal{U}\ T)))$$

In TMA, Since the constraint is a latency between AOP and $Match$, it can be easily written in TTL as:

$\mathcal{L}(\langle AOP, 2.5\ V, \nearrow\rangle, \langle Match, 2.5\ V, \nearrow\rangle, 10^{-4}) < t_{action}$ [3] .

[3] $t_{action} = 6s$ in this scenario.

The level threshold, 2.5 V, is the threshold to detect *true* or *false* on the Boolean signal (0 V and 5 V correspond to $false$ and $true$, respectively).

In order to compare three methods, the flying paster application has been implemented in 6 different scenarios in which the linear velocity of Active roll (v_A) is different. Table 7.3 listed those 6 scenarios where the linear speed of active roll, the time of AOP to *match* (t_{action}) and time to *contact* ($t_{termination}$) varies. When v_A is decreased, the time between events *AoP* and *match* (t_{action}), and also between *AoP* and *cut* ($t_{termination}$) will be obviously increased.

The required area for implementing the monitoring system is shown in figure 7.11.

Table 7.3: Six Different Scenarios for Flying Paster Application.

	A	B	C	D	E	F
v_A	22 m/s	20 m/s	18 m/s	16 m/s	14 m/s	12 m/s
t_{action}	2 s	3 s	4 s	5 s	6 s	7 s
$t_{termination}$	3 s	4 s	5 s	6 s	7 s	8 s

As Figure 7.10 depicts, conventional and Jakšić methods required more FFs and LUTs in the case study. With increasing the intervals, the FF and LUT utilization increases for the Jakšić method as well. In contrast, TMA takes a constant amount of memory in all scenarios because it does not require retention of signal history. When a signal event is observed, the result can be deduced. Moreover, the computation part – that affects the LUT size – is minimal by reducing operators (either event-based or level-based) to simple computations.

Low-Variability Signals

[Maler et al., 2007] proposes a metric for a Boolean signal φ called *variability* where it defines the number of value changing of φ in bounded time. φ is (Δ, n)-bounded variability if the number of changes in the value of φ is at most n. [Jakšić*et al.*, 2015]

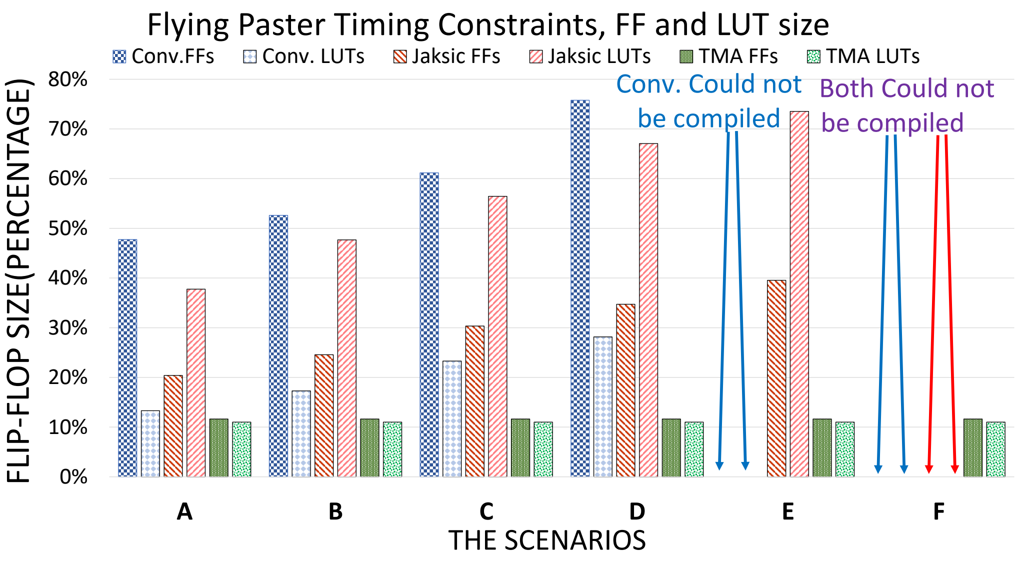


Figure 7.10: Comparison of FF and LUT Numbers in 3 Implemented Methods.

demonstrated that their approach is a good fit for monitoring of low-variability signals since it reduces the number of counters for memorizing the signal values. One of the timing specification of flying paster ($\Diamond_{[t_{AOP},t_{AOP}+t_{termination}]}(\uparrow cut)$) was implemented in three methods to see the efficiency of TMA in monitoring low-variability signals for different values of $t_{termination}$ as shown in the third row of Table 7.3. Table 7.3 shows 6 different values for the linear velocity of active roll (v_A) and the corresponding t_{action} and $t_{termination}$, which are used to determine the required area for implementing the monitoring system using conventional method, Jakšić approach, and TMA.

Figure 7.11 compares the FF utilization based upon the conventional, Jakšić, and TMA approaches for constraint evaluation in the case study application, where TMA used the least amount of memory.

Although Jakšić is better than conventional approach, TMA is the best among three technique. The reason is that not only TMA does not need counter or register and its calculations are lightweight but also that the constraint can be expressed using just one latency operator (between AOP and Cut), while expressing it in STL requires 17 and 18 operators in conventional and Jakšić, respectively.

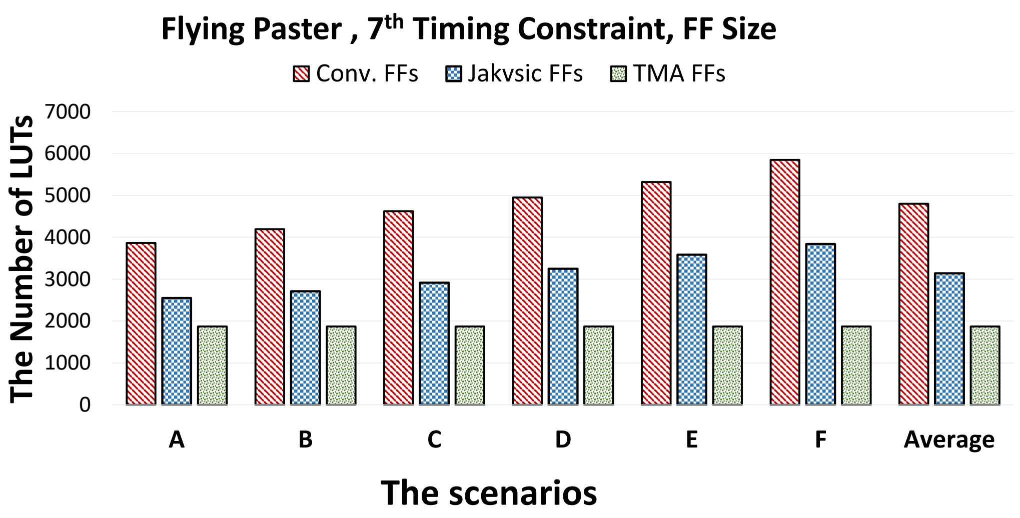


Figure 7.11: #FFs Utilization in Three Methods in Different Six Scenarios in Table 7.3

7.5.2 Monitoring Synchrophasor Using TMA

to the testing framework, sensors on both motors are monitored, and event timestamps are sent to the LabVIEW 2015 application core. The application is executed on a 64-bit Windows 7 with Intel(R) Core TM i7 2.93 GHz and 8 GB of RAM. In this case study, two 1024 data size Direct Memory Access (DMA) First-In-First Out (FIFO) data transfer with *64-bit quad signed integer* are used with time measurement precision on the order of 25 ns. This is because the cRIO NI-9067 works with a 40 MHz clock. The FPGA synthesis report indicated the Total Used Slices as: 16.9 % (2,257 out of 13,300). Here, a large size buffer for the DMA FIFO on the FPGA is allocated to the monitoring logic to reduce the communication and processing overhead on the desktop computer because smaller buffer sizes can cause loss of data. The TTL analysis is done on a desktop computer by gathering all timestamps from the cRIO devices.

The cRIO controller is managed by the LabVIEW tool containing a front panel interface. The front panel includes controls and indicators to send commands and

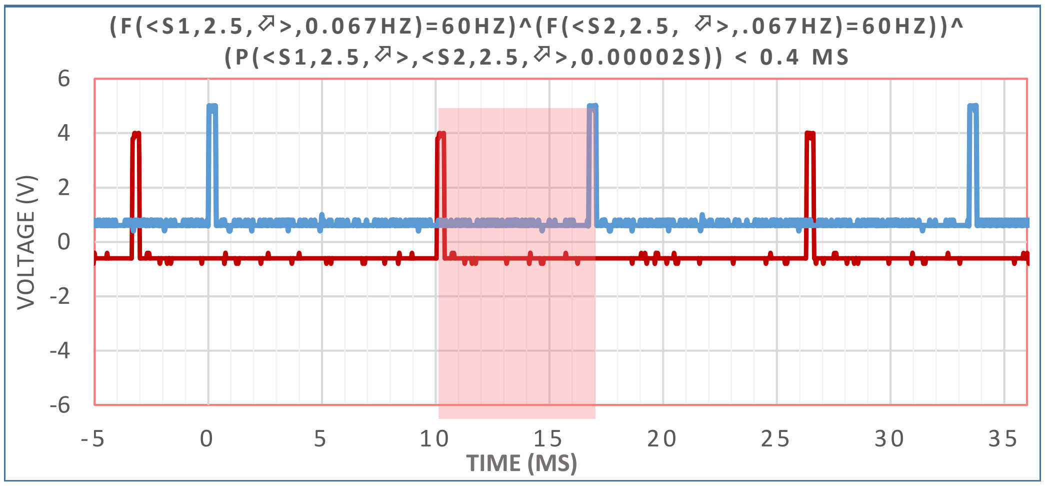


Figure 7.12: Acquired Results from Oscilloscope for the Synchrophasor.

monitor the parameters. In the testbed implementation, using the LabVIEW front panel shows the result of meeting the time constraint of the motor synchronization scenario online. Moreover, in order to verify the phase constraint, it is also validated using an oscilloscope (figure 7.12). The red and blue signals are acquired from optical sensors on master and slave motors from the logged data on oscilloscope. The transparent red area shows the time interval that the constraint is not met. The testing methodology has the capability to test the constraints by logging signal timestamps then applying the testing approach in an offline manner, but here the online monitoring has been implemented.

7.5.3 Monitoring Simultaneous Image Capturing Using TMA

The monitoring of the TTL statements showed that the latency difference in the actuation times of cameras was 0.01 s, as well as the data monitored on the oscilloscope (figure 7.13.a. Similarly, the interval between s_1 and s_2 is less than 0.2 s that is validated by the oscilloscope results depicted in figure 7.13.b. Therefore, the constraint is met on the deployed platforms and the Boolean indicator shows the constraint has been satisfied.

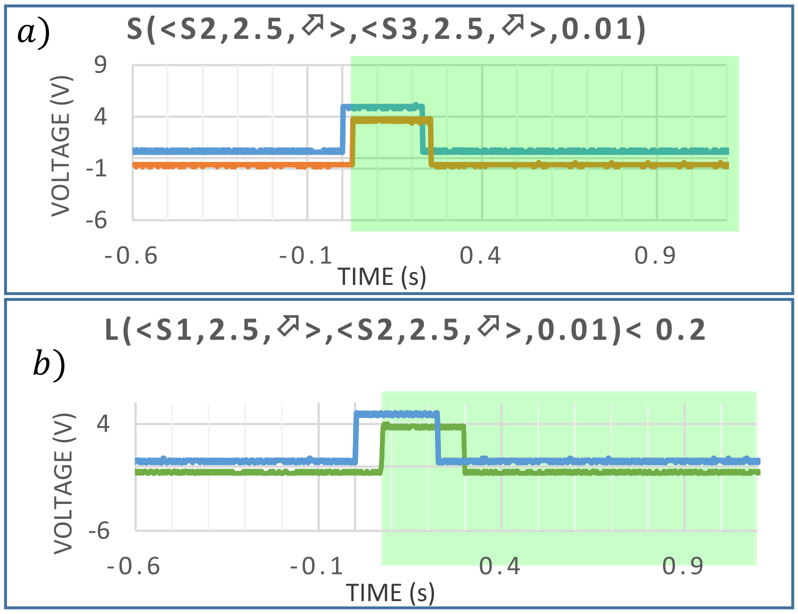


Figure 7.13: The Results for Image Capturing Acquired from Oscilloscope. a) Images Are Taken Simultaneously. b) The Delay between the Time That the Images Are Taken Is Less Than 0.2 s.

In this case study, three DMA FIFO memory buffers are used on FPGA for processing and Total Slices: 28.9 % (3,849 out of 13,300) of cRIO-9067 FPGA was used. Figure 7.13 shows the two trigger signals monitored on the oscilloscope.

7.6 Using Timestamp-based Monitoring Approach (TMA) to Monitor Temporal Behavior of Globally and Flying Paster Application

In this section, there are three different results for the implemented experiments. Firstly, the required FPGA area for a single operator (Globally) implemented using TMA is compared with the state of the art [Jakšić*et al.*, 2015], in FPGA implementation, in table 7.4. As the second part, some timing requirements of flying paster has been implemented and to show the efficiency of TMA, the occupied FPGA space is illustrated in two figures.

7.6.1 Efficiency of TMA in Implementing a Single Operator

In table 7.4, one single **Globally** operator with different intervals (from $[0, 100]$ to $[300, 600]$) was implemented on an NI cRIO FPGA board. The FPGA board is an NI-cRIO 9035 equipped with an on-board FPGA, Xilinx Kintex-7 7K70T, containing 82,000 FFs and 41,000 LUTs with a 40 MHz clock frequency. The table illustrates that in previous method Jakšić*et al.* [2015], the required area on FPGA depends on the length of the interval where TMA needs small and fixed size for all operators. #FF and #LUTs are the number of required flip-flops and lookup tables respectively. By increasing the length of the interval, the needed FFs and LUTs are increased in Jakšić where they are constant in TMA.

Table 7.4: Memory Requirement on FPGA for *Globally* Operator.

		#FFs		#LUTs	
		Jakšić [Jakšić*et al.*, 2015]	TMA	Jakšić [Jakšić*et al.*, 2015]	TMA
1	$\square_{[0,100]}$	1902	1820	2981	2696
2	$\square_{[0,200]}$	3935		5895	
3	$\square_{[0,300]}$	7821		9314	
4	$\square_{[150,200]}$	1891		2875	
5	$\square_{[300,400]}$	3702		5431	
6	$\square_{[450,600]}$	6312		9612	

7.7 The Satisfaction of the Testbed Requirements

In order to test and verify the timing constraints of the experimental setups, testbed timing specifications (synchronization accuracy, ADC sampling rate, ADC resolution, etc.) must exceed the CPS specifications as mentioned in section 6.5. In this section, the specifications of the monitoring equipment are scrutinized to know whether they are qualified for monitoring such CPS.

Table 7.5: The Specifications of Monitoring Devices

Monitoring Setup	DAQ card	Clk Drift	r_{sync}	Sampling Rate	ADC	Z_{in}	ϵ_{total}
cRIO-9067	NI-9381	5 PPM	1 Hz	10 kS/s	12 Bits	$1M\Omega$	$106\mu s$
cRIO-9035	NI-9232	5 PPM	1 Hz	102.4 kS/s	24 Bits	$305k\Omega$	$15.7\mu s$
ZYNQ 7000	onboard	5 PPM	1 Hz	100 kS/s	12 Bits	$10k\Omega$	$215\mu s$

7.7.1 NI-cRIO Setup

In a CompactRIO system (NI-cRIO), a controller with a processor and user-programmable FPGA is populated with one or more conditioned I/O modules from NI or third-party vendors. These modules provide direct sensor connectivity and specialty functions. cRIO is available in both a rugged industrial form factor and board-level design and it provides high-performance processing capabilities, sensor-specific conditioned I/O, and a closely integrated software toolchain that make them ideal for Industrial Internet of Things (IIoT), monitoring, and control applications.

As one of the testbeds for monitoring the time sensitive applications, two cRIO devices, NI-9067 and NI-9035, have been used as chassis while two signal acquisition modules, NI-9381 and NI-9232, installed on them for data acquisitions. Indeed, the modules collected the data to be processed on the FGPA board on the chassis. The specifications of the testbed equipment for NI-cRIO setup are summarized in the first two rows of table 7.5.

The cRIO FPGA borad has a clock with 40 MHz frequency and 5 x 10^{-6} clock drift. FPGA clocks are synchronized once a second using NI-TimeSync[Tim, 2010] that supports PTP. Measurement devices are connected via the dedicated Ethernet network. Implementation of the IEEE 802.1AS includes a very specific profile of IEEE 1588 (PTP) (part of IEEE 802.1 Time Sensitive Networking (TSN) standards) and uses hardware timestamping and compensation both in network elements and endpoints to minimize time synchronization errors. TSN generally provides both synchronization and also small and deterministic packet latency between testbed devices.

Table 7.6: The Specifications of Monitored CPS Applications.

Application	**Minimum** ε	Z_{out}	**Testbed**
Flying Paster	$100\mu s$	$25m\Omega$	cRIO-9067,9035,NI-9381 NI-9232
Synchrophasor	$80\mu s$	$25m\Omega$	cRIO-9067,9035,NI-9381 NI-9232
Simultaneous Image Capturing	$100\mu s$	470Ω	cRIO-9067,9035,NI-9381 NI-9232
RIM	$1ms$	50Ω	ZYNQ-7000

Clock Specifications

The clock drift of the measurement nodes is 5 x 10^{-6}, where each node synchronizes every second via PTP with a precision of $1\mu s$ to the grandmaster. The worst-case clock time offset of each cRIO is $\frac{5\ \mu s}{1\ s} + 1\mu s = 6\mu s$.

ADC and Sampling Time

Since the voltage range of digital module is from 0 V to 5 V and it uses a 12-bit and 24-bit ADC for NI-9381 and 9232 respectively, the ADC resolution V_{ADC} can be calculated as $\frac{5-0}{2^{12}} \approx 1$ mV and $\frac{5-0}{2^{24}} \approx 300$ nV in order.

As table 7.5 shows, the sampling rates for NI-9381 and NI-9232 modules are 20 and 102.4 kilo samples per second meaning the sampling time is around $50\mu s$ and $9.7\mu s$ respectively.

Based on the calculated errors in clock and ADC, the total error cRIO setup is:
NI-9381:

$$\epsilon_{total} = 6\mu s + 100\mu s = 106\mu s$$

NI-9232:

$$\epsilon_{total} = 6\mu s + 9.7\mu s = 15.7\mu s$$

7.7.2 ZYNQ 7000 Setup

PYNQ [PYN] is an open-source project from Xilinx that makes it easy to design embedded systems with Zynq Systems on Chips (SoCs). This framework enables embedded programmers to exploit the capabilities of Xilinx Zynq. In order to monitor RIM scenario, a PYNQ-Z1 board has been used for each vehicle. For each board, a Canakit Raspberry Pi Wifi Wireless Adapter/dongle (802.11 N/g/b 150) has been used for the wireless connectivity between subsystems in monitoring devices. For each event on CAVs, a specific pin has been connected to a pin on PYNQ board. The required electricity power provided by a Venom Lipo battery and all boards were synchronized using NTP every 1 second.

Clock Specifications

The clock drift of the measurement nodes is 5 x 10^{-6} in the clock frequency of $50MHz$. Each node synchronizes every second via NTP with a precision of $200\mu s$ to the grandmaster which is one of the CAVs. The worst-case clock time offset of each cRIO is $\epsilon_{wcco} = \frac{5\ \mu s}{1\ s} + 200\mu s = 205\mu s$.

ADC and Sampling Time

Since the voltage range of digital module is from 0 V to 3.3 V and it uses a 12-bit. The ADC resolution V_{ADC} can be calculated as $\frac{5-0}{2^{12}} \approx 1$ mV. As table 7.5 depicts, the sampling rates for ZYNQ-7000 is 100 kilo samples per second meaning the sampling time is around $10\mu s$. Therefore, $\epsilon_{ADC} = 10\mu s$.

Based on the calculated errors in clock and ADC, the total error for ZYNQ setup is:

$$\epsilon_{total} = 205\mu s + 10\mu s = 215\mu s$$

7.7.3 Testbed Capability Analysis

In this section, it is shortly shown that whether the testbed is qualified for the monitoring the applications.

Table 7.6 summarizes the specifications of applications while table 7.5 is demonstrating the characteristics of monitoring devices for monitoring the case studies. As the ϵ_{total} column shows, the precision of NI-9381 is not enough (it is $106\mu s$) while NI-9232 (its precision is $15.7\mu s$) is good enough for testing the applications ($\varepsilon > \epsilon_{total}$).

In Robust Intersection Manager, the smallest tolerable error is $1ms$ and since the total error in ZYNQ is $215\mu s$, PYNQ board is qualified to monitor RIM.

About input impedance of the monitoring equipment, all of them are qualified for monitoring. Output impedance of the ArduCAM boards, for simultaneous image capturing, is 470 Ω and input impedance of the cRIO is $305k\Omega$. Therefore, the loading effect is very small and the testbed is suitable to verify the timing constraints. Similarly, in synchrophasor and flying paster, two optical sensors (Omron EE-SX970-C1) has at maximum $25m\Omega$ output impedance in comparison with $305k\Omega$. In RIM scenario, each pin of ESP 8266 board has 50Ω of resistors that is too small when they are connected to a PYNQ board whose pins have a resistor of $10k\Omega$.

Aviral Shrivastava *et al.* A testbed to verify the timing behavior of cyber-physical systems. In *DAC 2017*. ACM, 2017.

MohammadReza Mehrabian, Saadat Pour Mozafari, and Behrouz Zolfaghari. An approach to exploiting proper multiples of the generator polynomial in parallel crc computation. In *2012 IEEE International Conference on Computer Science and Automation Engineering (CSAE)*, volume 1, pages 614–617. IEEE, 2012.

Alexander Bolotov, Artie Basukoski, Oleg Grigoriev, and Vasilyi Shangin. Natural deduction calculus for linear-time temporal logic. In *European Workshop on Logics in Artificial Intelligence*, pages 56–68. Springer, 2006.

Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):225–299, 1990a.

Oded Maler and Dejan Ničković. Monitoring properties of analog and mixed-signal circuits. *International Journal on Software Tools for Technology Transfer*, 15(3): 247–268, 2013a. ISSN 14332779. doi: 10.1007/s10009-012-0247-9.

Aviral Shrivastava, Mohammadreza Mehrabian, Mohammad Khayatian, Patricia Derler, Hugo Andrade, Kevin Stanton, Ya Shian Li-Baboud, Edward Griffor, Marc Weiss, and John Eidson. INVITED: A Testbed to Verify the Timing Behavior of Cyber-Physical Systems: Invited. *Proceedings - Design Automation Conference*, Part 12828(69):1–6, 2017. ISSN 0738100X. doi: 10.1145/3061639.3072955.

W Chipman, C Grimm, and C Radojicic. Coverage of Uncertainties in Cyber-Physical Systems. *ZuE 2015; 8. GMM/ITG/GI-Symposium Reliability by Design; Proceedings of*, 3:1–8, 2015.

Tao Ma, Shaukat Ali, Tao Yue, and Maged Elaasar. Testing self-healing cyber-physical systems under uncertainty : a fragility-oriented approach. *Software Quality Journal*, 27:615–649, 2019.

Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: a conceptual model. In *European conference on modelling foundations and applications*, pages 247–264. Springer, 2016a.

Carna Radojicic, Christoph Grimm, Axel Jantsch, and Michael Rathmair. Towards verification of uncertain cyber-physical systems. *Electronic Proceedings in Theoretical Computer Science, EPTCS*, 247(Snr):1–17, 2017a. ISSN 20752180. doi: 10.4204/EPTCS.247.1.

Edward A. Lee. Cyber physical systems: Design challenges. *Proceedings - 11th IEEE Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing, ISORC 2008*, 11:363–369, 2008. doi: 10.1109/ISORC.2008.25.

Man Zhang, Bran Selic, Shaukat Ali, Tao Yue, Oscar Okariz, and Roland Norgren. Understanding uncertainty in cyber-physical systems: A conceptual model. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9764:247–264, 2016b. ISSN 16113349. doi: 10.1007/978-3-319-42061-5{_}16.

Peter B. Ladkin. A320-211 warsaw accident report. `http://sunnyday.mit.edu/accidents/warsaw-report.html`, 3 1994. (Accessed on 11/16/2020).

US GAO. Software Problem Led to System Failure at Dhahran, Saudi Arabia. *US GAO Reports, report no. GAO/IMTEC-92-26*, 1(1):1–20, 1992. URL `http://www.gao.gov/assets/220/215614.pdf%5Cnhttp://www.gao.gov/products/IMTEC-92-26`.

Robert N. Charette. Nissan recalls nearly 1 million cars for air bag software fix - ieee spectrum. `https://spectrum.ieee.org/riskfactor/transportation/safety/nissan-recalls-nearly-1-million-cars-for-airbag-software-fix`, 03 2014. (Accessed on 11/17/2020).

Yashwanth Annpureddy *et al.* S-TaLiRo: A Tool for Temporal Logic Falsification for Hybrid Systems. In *TACAS*. Springer, 2011.

Mohammadreza Mehrabian and other. Timestamp temporal logic (ttl) for testing the timing of cyber-physical systems. In *TECS 2017*. ACM, 2017.

Naeem Esfahani and Sam Malek. Uncertainty in self-adaptive software systems. In *Software Engineering for Self-Adaptive Systems II*, pages 214–238. Springer, 2013.

John W van de Lindt, BR Ellingwood, Paolo Gardoni, and DT Cox. Modeling community resilience to earthquakes and tsunamis: An overview of the center for risk-based community resilience planning. In *11th National Conference on Earthquake Engineering 2018: Integrating Science, Engineering, and Policy, NCEE 2018*, pages 3694–3698. Earthquake Engineering Research Institute, 2018.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a Formal Model of Safe and Scalable Self-driving Cars. *arXiv*, 1(1):1–37, 2017.

Mohammad Khayatian, Mohammadreza Mehrabian, Harshith Allamsetti, Kai-Wei Liu, Po-Yu Huang, Chung-Wei Lin, and Aviral Shrivastava. Cooperative driving of connected autonomous vehicles using responsibility-sensitive safety (rss) rules. In *Proceedings of the ACM/IEEE 12th International Conference on Cyber-Physical Systems*, pages 11–20, 2021.

Antoine Cailliau and Axel Van Lamsweerde. Handling knowledge uncertainty in risk-based requirements engineering. In *2015 IEEE 23rd International Requirements Engineering Conference (RE)*, pages 106–115. IEEE, 2015.

William L Oberkampf, Sharon M DeLand, Brian M Rutherford, Kathleen V Diegert, and Kenneth F Alvin. Error and uncertainty in modeling and simulation. *Reliability Engineering & System Safety*, 75(3):333–357, 2002.

Roger G Ghanem and Steven F Wojtkiewicz. Special issue on uncertainty quantification. *SIAM Journal on Scientific Computing*, 26(2):vii–vii, 2004.

P Ciffroy and M Benedetti. A comprehensive probabilistic approach for integrating natural variability and parametric uncertainty in the prediction of trace metals speciation in surface waters. *Environmental Pollution*, 242:1087–1097, 2018.

P Ciffroy. A comprehensive probabilistic approach for integrating and separating natural variability and parametric uncertainty in the prediction of distribution coefficient of radionuclides in rivers. *Journal of Environmental Radioactivity*, 225: 106371, 2020.

John P Davis and Jim W Hall. A software-supported process for assembling evidence and handling uncertainty in decision-making. *Decision Support Systems*, 35(3): 415–433, 2003.

Stephen A Edwards and Edward A Lee. The case for the precision timed (pret) machine. In *Proceedings of the 44th annual Design Automation Conference*, pages 264–265, 2007.

Janos Sztipanovits and Gabor Karsai. Model-integrated computing. *Computer*, 30 (4):110–111, 1997.

Peter Hehenberger, Birgit Vogel-Heuser, David Bradley, Benoît Eynard, Tetsuo Tomiyama, and Sofiane Achiche. Design, modelling, simulation and integration of cyber physical systems: Methods and applications. *Computers in Industry*, 82: 273–289, 2016.

Github - apolloauto/apollo: An open autonomous driving platform. `https://github.com/ApolloAuto/apollo`, 06 2021. (Accessed on 06/22/2021).

Sadaf Mustafiz, Cláudio Gomes, Hans Vangheluwe, and Bruno Barroca. Modular design of hybrid languages by explicit modeling of semantic adaptation. In *2016 Symposium on Theory of Modeling and Simulation (TMS-DEVS)*, pages 1–8. IEEE, 2016.

Andreas Tolk, Ernest H Page, and Saurabh Mittal. Hybrid simulation for cyber physical systems: state of the art and a literature review. In *SpringSim (ANSS)*, pages 10–1, 2018.

Carna Radojicic, Christoph Grimm, Axel Jantsch, and Michael Rathmair. Towards verification of uncertain cyber-physical systems. *arXiv preprint arXiv:1705.00519*, 2017b.

Amir Pnueli. The temporal logic of programs. *Proceedings - Annual IEEE Symposium on Foundations of Computer Science, FOCS*, 1977-Octob:46–57, 1977. ISSN 02725428. doi: 10.1109/sfcs.1977.32.

Ron Koymans. Specifying real-time properties with metric temporal logic. *Real-time systems*, 2(4):225–299, 1990b.

Oded Maler and Dejan Ničković. Monitoring properties of analog and mixed-signal circuits. *International Journal on Software Tools for Technology Transfer*, 15(3): 247–268, 2013b. ISSN 14332779. doi: 10.1007/s10009-012-0247-9.

Air bags — national highway traffic safety administration (nhtsa). `https://one.nhtsa.gov/Laws-&-Regulations/Air-Bags`, 06 2021. (Accessed on 05/03/2021).

Xi Zheng and Christine Julien. Verification and validation in cyber physical systems: research challenges and a way forward. In *SEsCPS 2015*, pages 15–18. IEEE, 2015.

Charles Antony Richard Hoare. An axiomatic basis for computer programming. *Communications of the ACM*, 12(10):576–580, 1969.

Amir Pnueli and Yonit Kesten. A deductive proof system for ctl. In *International Conference on Concurrency Theory*. Springer, 2002.

Zhenhua Duan and Nan Zhang. A complete axiomatization of propositional projection temporal logic. In *TASE'08*. IEEE, 2008.

Oded Maler and Dejan Nickovic. Monitoring Temporal Properties of Continuous Signals. *Springer*, pages 152–166, 2004. ISSN 03029743. doi: 10.1007/978-3-540-30206-3{_}12. URL `http://link.springer.com/10.1007/978-3-540-30206-3_12`.

James F Allen and George Ferguson. Actions and Events in Interval Temporal Logic. *Journal of Logic and Computation*, pages 1–56, 1994.

Dag Prawitz. *Natural deduction: A proof-theoretical study.* Courier Dover Publications, 2006.

Oded Maler and other. Monitoring properties of analog and mixed-signal circuits. *International Journal on Software Tools for Technology*, 2013.

Ali Abbass Zoraghchian Moslem Didehban and Mohammadreza Mehrabian. A fault detection method for combinational circuits. *International Journal of Advanced Network, Monitoring and Controls*, pages 1–5, 2108.

Stefan Jakšić*et al.* From Signal Temporal Logic to FPGA Monitors. In *MEMOCODE*, 2015.

NIST Time and Frequency from A to Z Glossary. `https://www.nist.gov/time-and-frequency-services/d`, 2017. [Online; Accessed: 2017-03-21].

John C Eidson and Kevin B Stanton. Timing in Cyber-Physical Systems: The Last Inch Problem. In *Precision Clock Synchronization for Measurement, Control, and Communication (ISPCS), 2015 IEEE International Symposium on*, pages 19–24. IEEE, 2015.

David L Mills. RFC 1305: Network Time Protocol (Version 3) Specification. *Implementation and Analysis*, 1992.

IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems. *IEEE Std 1588-2008 (Revision of IEEE Std 1588-2002)*, pages 1–269, July 2008. doi: 10.1109/IEEESTD.2008.4579760.

Maciej Lipiński, Tomasz Włostowski, Javier Serrano, and Pablo Alvarez. White Rabbit: A PTP Application for Robust Sub-nanosecond Synchronization. In *ISPCS*, pages 25–30. IEEE, 2011.

Mohammad Khayatian, Mohammadreza Mehrabian, Edward Andert, Rachel Dedinsky, Sarthake Choudhary, Yingyan Lou, and Aviral Shirvastava. A survey on intersection management of connected autonomous vehicles. *ACM Transactions on Cyber-Physical Systems*, 4(4):1–27, 2020a.

Rachel Dedinsky et al. A dependable detection mechanism for intersection management of connected autonomous vehicles. In *ASD*, 2019.

Mohammad Khayatian, Rachel Dedinsky, Sarthake Choudhary, Mohammadreza Mehrabian, and Aviral Shrivastava. R 2 im-robust and resilient intersection management of connected autonomous vehicles. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020b.

Mohammad Khayatian, Aviral Shrivastava, and Mohammadreza Mehrabian. Systems and methods for intersection management of connected autonomous vehicles, May 28 2020c. US Patent App. 16/694,451.

Mohammad Khayatian *et al.* Rim: Robust intersection management for connected autonomous vehicles. In *RTSS*. IEEE, 2018.

Shalev-Shwartz Shai and other. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

D Mills. Network time protocol (version 2) specification and implementation; rfc-1119. *Internet Requests for Comments*, (1119), 1989.

IEEE Instrumentation and Measurement Society. IEEE 1588 Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems (IEEE Std 1588-2002). 2002.

Frequency excursions. `https://www.nerc.com/pa/RAPA/PA/Pages/FrequencyExcursions.aspx`. (Accessed on 08/05/2020).

Oded Maler, Dejan Nickovic, and Amir Pnueli. On synthesizing controllers from bounded-response properties. In *International Conference on Computer Aided Verification*, pages 95–107. Springer, 2007.

NI Time Sync, Version 1.1. `http://www.ni.com/pdf/manuals/373185a.pdf`, 2010. [Online; 2010 National Instruments Corporation].

Pynq - python productivity for zynq - home. `http://www.pynq.io/`. (Accessed on 08/04/2020).

Yu Jiang, Mingzhe Wang, Xun Jiao, Houbing Song, Hui Kong, Rui Wang, Yongxin Liu, Jian Wang, and Jiaguang Sun. Uncertainty theory based reliability-centric cyber-physical system design. In *2019 International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, pages 208–215. IEEE, 2019.

Baoding Liu. Uncertainty theory. In *Uncertainty theory*, pages 205–234. Springer, 2007a.

Jinjing Huang, Enda Howley, and Jim Duggan. The ford method: A sensitivity analysis approach. In *Proceedings of the 27th International Conference of the System Dynamics Society, Albuquerque, USA, The System Dynamics Society. 6Other models with the loop picker algorithm are available from the author by email*, 2009.

Piroska Haller and Béla Genge. Using sensitivity analysis and cross-association for the design of intrusion detection systems in industrial cyber-physical systems. *IEEE Access*, 5:9336–9347, 2017.

Elena Lisova, Elisabeth Uhlemann, Johan Åkerberg, and Mats Björkman. Monitoring of clock synchronization in cyber-physical systems: A sensitivity analysis. In *2017 international conference on Internet of Things, embedded systems and communications (IINTEC)*, pages 134–139. IEEE, 2017.

Baoding Liu. *Uncertainty Theory*. Springer Publishing Company, Incorporated, 2nd edition, 2007b. ISBN 3540731644.

APPENDIX A

UNCERTAINTY ANALYSIS

In the system modeling, if asked how the system would operate with inputs similar to those in the historical database, the model should be able to interpolate within the available knowledge about the target system. Indeed, it should provide a fairly accurate estimation and we know that estimation always has a level of error since the model's ability to reproduce current and recent operations is not perfect. In this regard, an uncertainty analysis needs to consider how well a model can replicate current operations, and how similar the target conditions or scenarios are to those described in the historical record. In this section, we first define some preliminary concepts about uncertainty analysis and then explain Uncertainty theory, Probability theory, sensitivity and uncertainty analysis and, a comparison between them.

A.0.1 Uncertainty Theory

Uncertainty theory is used to study uncertain phenomenon that are usually caused by randomness, fuzziness or uncertainty. In fact, it is a way to measure the likelihood of an event being true. The uncertainty theory is developed based on four axioms: i) normality, ii) monotonicity, iii) self-duality, and iv) countable subadditivity. Note that we use $M\{\Lambda\}$ to represent an "uncertain measure". Given Γ is a non-empty set, L be a σ-algebra over Γ, $\Lambda \in L$ be an event, and $M\{\Lambda\}$ be the level Λ occurs, we have the following axioms:
Axiom 1. *Normality*: $M\{\Lambda\} = 1$, stating that the occurrence level for the complete set is 1 (always).
Axiom 2. *Monotonicity*: $M\{\Lambda_1\} \leq M\{\Lambda_2\}$ whenever $\Lambda_1 \subset \Lambda_2$, saying that the occurrence level of a set is always greater than or equal to its subsets.
Axiom 3. *Duality*: $M\{\Lambda\} + M\{\Lambda^c\} = 1$, meaning that the occurrence level of an event Λ and its complement Λ^c is 1.
Axiom 4. *Countable Subadditivity*: for every countable sequence of event:
Λ_i $M\{\bigcup_{i=1}^{\infty} \Lambda_i\} \leq \sum_{i=1}^{\infty} M\{\Lambda_i\}$
stating that the occurrence level of union of a sequence of countable events is smaller than or equal to the occurrence level of the summation of those events. Uncertainty theory has been used for modeling uncertainty in CPS Jiang et al. [2019] where an uncertain value with a distribution is considered.

Uncertain Variable: An uncertain variable η is a measurable function from an arbitrary uncertain space defined by (Γ, L, M) to the set of real numbers, that is, for any Borel set B of real numbers, the set $\{\eta \in B\} = \{\gamma \in \Gamma | \eta(\gamma)] \in B\}$ is an event Liu [2007a].

Uncertainty Distribution: The uncertainty distribution is a mapping as $\Phi : R \rightarrow [0, 1]$ of an uncertain variable η where $\Phi(x) = M\{\gamma \in \Gamma | \eta(\gamma) \leq x\}$. In simple words, the uncertainty distribution indicates how the occurrence frequency of an event is distributed.

A.0.2 Probability Theorem

Probability theory is used to study the behavior of random phenomena with the help of mathematics. The probability theory is defined through three axioms: 1)

Normality, 2) Nonnegativity, and 3) Countable Additivity.

Given Ω is a nonempty set, $\mathcal{A}$ is a σ-algebra over Ω, each element in $\mathcal{A}$ is called an event (A) and $Pr\{A\}$ indicates the probability that event A will occur. Now, following axioms should be held:

Axiom 1: *Normality:* $Pr(\Omega) = 1$, meaning the probability of occurrence of either of possible events is 1 (always).

Axiom 2: *Non-negetivity:* Pr(A) ≥ 0 for any $A \in \mathcal{A}$, which indicates the probability of an event cannot be negative.

Axiom 3: *Countable Additivity:* For every countable sequence of mutually disjoint events A_i, $Pr\{ \cup_{i=1}^{\infty} A_i\} = \sum_{i=1}^{\infty} Pr\{A_i\}$.

Probability Distribution The probability distribution of a random variable η is a mapping $\Phi : \mathcal{R} \rightarrow [0, 1]$ and $\Phi(x) = Pr\{\omega \in \Omega | \eta(\omega) \leq x\}$ indicates the probability that the random variable η is less than or equal to x.

A.0.3 Sensitivity Analysis Vs Uncertainty Analysis

In this section, we study two approaches: i) Sensitivity analysis, which studies the relative importance of different input factors on the model output and ii) Uncertainty analysis, which studies just the quantification of uncertainty in model output. In the CPS domain, uncertainty and sensitivity analysis are employed to predict the behavior of the system and determine the future outcome of the system.

Sensitivity Analysis

Sensitivity analysis is done to measure the sensitivity of a model to a parameter. In other words, how the model changes with the change in the parameters. One way to determine the sensitivity of a parameter is to compute the contribution of the variable of interest. The simplest approach is to vary uncertain variables within their reasonable range, one at a time, and then compute the corresponding output. This approach is widely employed in engineering economics. In the sensitivity coefficient method, the sensitivity index SI is computed as:

$$SI = \frac{f(u_0 + \Delta u) - f(u_0 - \Delta u)}{2\Delta u} \tag{A.1}$$

where u_0 is the input base value, Δu is the changes in the value of input and $f(u)$ is the output for the given input u.

Ford method Huang et al. [2009] is another method to determine how sensitive is a variable of interest in the model. In this approach, the error between the variable of interest throughout the time v_i^t and the variable of interest in the reference model v_0^t are calculated as:

$$\Delta v_i^t = v_i^t - v_0^t \tag{A.2}$$

and the variance is computed as:

$$S_i^t = (\Delta v_i^t)^2$$

Then, the contribution of each experiment (influence factor) is computed as:

$$c_i^t = sqrt\frac{S_i^t}{\sum_{m=1}^{n} S_m^t}$$

finally, the re-scaled contribution of each experiment is computed as:

$$r_i^t = \frac{c_i^t}{\sum_{m=1}^{n} c_m^t}$$

Similarly, standard deviation has also been used to compute a sensitivity index for attack detection Haller and Genge [2017]:

$$c_i = \frac{std(v_i)}{std(v_0)}$$

Sometimes errors in two or more variable simultaneously contribute to the output's uncertainty which requires using joint probabilistic distributions. The first-order approach computes the sensitivity based on a multivariate linear analysis for a given performance indicator $I = P(f(u))$. In this approach, expected value and variance are computed as:

$$E[I] = P(\bar{u}) + \frac{1}{2}\left\{ \sum_i \sum_j [\frac{\partial P^2}{\partial u_i \partial u_j}] COV(u_i, u_j) \right\} \tag{A.3}$$

and

$$Var[I] = \sum_i \sum_j [\frac{\partial P}{\partial u_i}\frac{\partial P}{\partial u_j}] COV(u_i, u_j) \tag{A.4}$$

where $\bar{u}$ is the mean value of input parameters and $COV(u_i, u_j)$ is the covariance of two parameter u_i and u_j which is computed as $COV(u_i, u_j) = E\big[u_i - E[u_i]\big]E\big[u_j - E[u_j]\big]$. It should be noted that first-order uses Taylor approximation and therefore may not be very accurate for non-linear systems. Sensitivity analysis has been used to perform decision making under uncertain conditions, for example, to detect if an attack has happened Lisova et al. [2017]. This is mainly done by computing the variance of the uncertain variable.

Uncertainty Analysis

The uncertainty analysis tries to guess the set of possible outputs and their probability. To do so, a Probability Density Function (PDF) is used which describes the occurrence probability of each value. Figure A.1 shows two examples of a PDF as a continuous and a discrete function. One step in uncertainly analysis is to identify all sources of uncertainty that can contribute to the input/outputs joint probability distribution. Then, the mean and standard deviation can be estimated for the output. Using probability-based approaches, one can determine the other performance measures such as the probability that the value exceeds an amount.

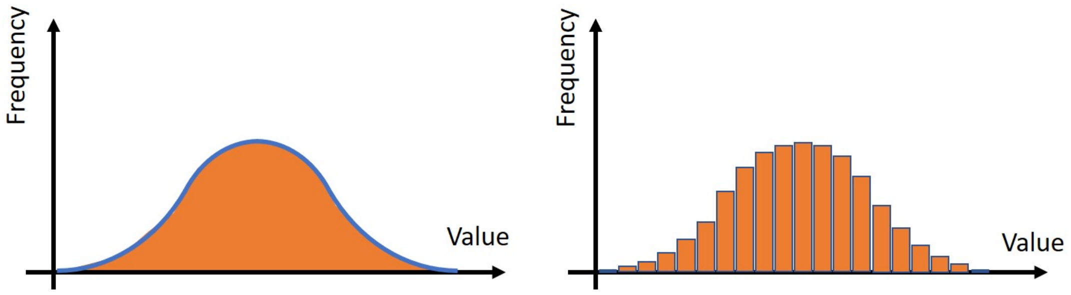


Figure A.1: PDF of a Gaussian distribution (continuous) and (discrete)

A.0.4 Uncertain Programming Model

As far as we know, no work deals with variables as ‘uncertain’ but rather but ‘stochastic’ or ‘arbitrary’. However, in reality, both approaches are improper to explain CPS’s behaviors without any information (or it is equivalent to Known-unknown knowledge.). To solve this problem, Liu [2007b] proposes multiple set of solutions for insufficient (or even no) samples, which are comprehensively called ‘Uncertain Programming’. Uncertain programming starts from this assumption: *’humans tend to over-estimate system’s danger’*. In other words, humans manually designate probability distributions (or belief degree in other term) and these cover all dangers even though it might be inefficient. Also it teaches us some ways to reasonably deal with a situation when given minimum set of data. Uncertain programming is a mathematical programming method involving uncertain variables to output a decision with minimum expected objective value subject to a set of chance constraints. Assuming that x is a decision vector, and ξ is an uncertain vector. The uncertain vector consists of multiple uncertain variables. Unlike typical linear programming problem, uncertain variables are non-deterministic and so is the objective function. However, we can minimize its expected value and it is called Expected Value Model (EVM).

$$\begin{cases} \min\limits_{x} E\left[f(x,\xi)\right] \\ \text{subject to:} \\ \mathcal{M}\left\{g_j(x,\xi) \le 0\right\} \ge \alpha_j, j = 1, 2, \cdots, p. \end{cases}$$

Definition 1 A vector x is called a **feasible solution** to the uncertain programming model if

$$\{\mathcal{M}\left\{g_j(x,\xi) \le 0\right\} \ge \alpha_j$$

for $j = 1, 2, \cdots, p$.

Definition 2 A feasible solution x^* is called an **optimal solution** to the uncertain programming model if

$$\left\{ \min_{x} E\left[f(x^*,\xi)\right] \le \min_{x} E\left[f(x,\xi)\right] \right.$$

for any feasible solution x.

APPENDIX B

THE PROOF FOR TMA ALGORITHMS AND TTL RULES

Lemma B.1.1. For a positive interval, I_i^+ defined on ψ, the eventually statement ($\Diamond_{[a,b]}\psi$) is true starting from start timestamp, $t_r^{\Diamond_{[a,b]}\psi}$, up to finish timestamp, $t_f^{\Diamond_{[a,b]}\psi}$, if the width of next negative interval (I_{i+1}^-) is greater than $b-a$ where:
$t_r^{\Diamond_{[a,b]}\psi} = t_r^\psi - b$ and $t_f^{\Diamond_{[a,b]}\psi} = t_f^\psi - a$

Proof. According to the eventually definition, $\Diamond_{[a,b]}\psi$ is true if there exists a time $t' \in [t+a, t+b]$ such that ψ is satisfied.

Assume t_r^ψ is the first time that ψ is true. Then, the following condition is always true:

$$t_r^{\Diamond_{[a,b]}\psi} + a < t_r^\psi < t_r^{\Diamond_{[a,b]}\psi} + b$$

Since we looking for the earliest time that the condition is met, the rising time of $\Diamond_{[a,b]}\psi$ is equal to: $t_r^{\Diamond_{[a,b]}\psi} = t_r^\psi - b$. Similarly, assume t_f^ψ is the last instance that ψ is true. Then,

$$t_f^{\Diamond_{[a,b]}\psi} + a < t_f^\psi < t_f^{\Diamond_{[a,b]}\psi} + b$$

is always true.

Since we are looking for the latest time that the condition is met, the falling time of $\Diamond_{[a,b]}\psi$ is equal to:

$$t_f^{\Diamond_{[a,b]}\psi} = t_f^\psi - a$$

Since the width of the next negative interval is greater than $b-a$, we can show that the computed rising timestamp corresponding to the next positive interval is not less than the falling timestamp corresponding to the current interval (i.e. $t_{f_i}^{\Diamond_{[a,b]}\psi} < t_{r_{i+1}}^{\Diamond_{[a,b]}\psi}$):

$$^{\psi}_{f_i} - a < t_{r_{i+1}}^\psi - b$$

is always true since the width of the next negative interval is greater than $b-a$ ($t_{r_{i+1}}^\psi - t_{f_i}^\psi > b-a$). □

Lemma B.1.2. For a positive interval, I_i^+ on ψ, the statement $\Box_{[a,b]}\psi$ is true starting from start timestamp, $t_r^{\Box_{[a,b]}\psi}$, up to finish timestamp, $t_f^{\Box_{[a,b]}\psi}$, if the width of I_i^+ is greater than $b-a$, where $t_r^{\Box_{[a,b]}\psi} = t_r^\psi - a$ and $t_f^{\Box_{[a,b]}\psi} = t_f^\psi - b$.

Proof. According to the globally definition, $\Box_{[a,b]}\psi$ is true if for all time instances $t' \in [t+a, t+b]$, ψ is satisfied. In order to find the rising timestamp of $\Box_{[a,b]}\psi$, we are looking for the first time that ψ is true.

Assume t_r^ψ is the first time that ψ is true. Then,

$$t_r^\psi < t_r^{\Box_{[a,b]}\psi} + a$$

is true.

Since we are looking for the earliest time, the rising time of $\Box_{[a,b]}\psi$ is equal to:

$$t_r^{\Box_{[a,b]}\psi} = t_r^{\psi} - a$$

.

Similarly, assume t_f^{ψ} is the last instance that ψ is true. Then,

$$t_f^{\psi} > t_f^{\Box_{[a,b]}\psi} + b$$

is true.

Since we are looking for the latest time, the falling time of $\Box_{[a,b]}\psi$ is equal to:

$$t_f^{\Box_{[a,b]}\psi} = t_f^{\psi} - b$$

.

Since the width of positive interval is greater than $b - a$, we can show that the computed falling timestamp is always greater than the falling one (i.e. $t_f^{\Box_{[a,b]}\psi} > t_r^{\Box_{[a,b]}\psi}$):

$$t_f^{\psi} - b > t_r^{\psi} - a$$

which is always true since the width of the positive interval is greater than $b - a$ ($t_r^{\psi} - t_f^{\psi} > b - a$). □

Definition B.1.1. Overlapped Intervals, A positive interval on a boolean signal, $I^+_{\psi_{1_i}}$ overlaps with another positive interval on another boolean signal, $I^+_{\psi_{2_j}}$ if and only if $\exists t' \in I^+_{\psi_{1_i}}, \exists t'' \in I^+_{\psi_{2_j}} s.t.\ (s, t') \models \psi_1 \wedge (s, t'') \models \psi_2 \wedge t' = t''$

Lemma B.1.3. An until statement ($\psi_1 \mathcal{U} \psi_2$) is false for non-overlapped intervals of ψ_1 and ψ_2.

Proof. If $I^+_{\psi_{1_i}}$ and $I^+_{\psi_{2_j}}$ does not overlap, it means there does not exist a time that both ψ_1 and ψ_2 are satisfied (i.e. $\nexists t' \in I^+_{\psi_{1_i}} \cup I^+_{\psi_{1_i}} s.t.\ (s, t) \models \psi_1 \wedge \psi_2$.)

According to until operator definition, there should be a point that ψ_2 is satisfied and ψ_1 is satisfied up to that point. Therefore, the result of evaluation is false if there does not exist a pair of intervals that overlap. □

Lemma B.1.4. Assume that I_1 and I_2 are two positive intervals defined on boolean signals ψ_1 and ψ_2 respectively. If the length of overlapping interval of I_1 and I_2 is greater than a (i.e. $I^+_{\psi_{1_i}} \cap I^+_{\psi_{2_j}} > a$), $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is true starting from $t_r^{\mathcal{U}}$ and up to $t_f^{\mathcal{U}}$ so, we have:

$$t_r^{\mathcal{U}} = max(t_{r_i}^{\psi_1}, t_{r_j}^{\psi_2} - b)$$

and

$$t_f^{\mathcal{U}} = min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$$

where

$t_{r_i}^{\psi_1}$ and $t_{r_j}^{\psi_2}$ are timestamps corresponding to rising edges of positive intervals of ψ_1 and ψ_2, and $t_{f_i}^{\psi_1}$ and $t_{f_j}^{\psi_2}$ are timestamps corresponding to falling edges of positive intervals of ψ_1 and, ψ_2 respectively.

Proof. To ensure $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is true starting from $t_r^{\mathcal{U}}$ until $t_f^{\mathcal{U}}$, three conditions must be satisfied:

i) computed $t_r^{\mathcal{U}}$ must be less than $t_f^{\mathcal{U}}$ to indicate a positive interval and

ii) $max(t_{r_i}^{\psi_1}, t_{r_j}^{\psi_2} - b)$ is the start time and iii) $min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_1}) - a$ is the finish time.

Part i) To prove the computed rising time is less than the falling one, we need to show that

$$t_r^{\mathcal{U}} < t_f^{\mathcal{U}}$$

or

$$max(t_{r_i}^{\psi_1}, t_{r_j}^{\psi_2} - b) < min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$$

Assuming

$$t_{r_i}^{\psi_1} > t_{r_j}^{\psi_2} - b$$

we have

$$t_{r_i}^{\psi_1} < min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$$

If $t_{f_i}^{\psi_1} < t_{f_i}^{\psi_2}$, it yields $t_{r_i}^{\psi_1} < t_{f_i}^{\psi_1} - a$

which is always true since the width of the positive interval of ψ_1 is greater than a.

If $t_{f_i}^{\psi_2} < t_{f_i}^{\psi_1}$, it yields $t_{r_i}^{\psi_1} < t_{f_i}^{\psi_2} - a$

which is always true since the width of overlapping interval of ψ_1 and ψ_2 is greater than a.

Assuming

$$t_{r_j}^{\psi_2} - b > t_{r_i}^{\psi_1}$$

we have

$$t_{r_j}^{\psi_2} - b < min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$$

If $t_{f_i}^{\psi_1} < t_{f_i}^{\psi_2}$, it yields

$$t_{r_j}^{\psi_2} - b < t_{f_i}^{\psi_1} - a$$

or

$$t_{f_i}^{\psi_1} - t_{r_j}^{\psi_2} > a - b$$

It's obvious that $a - b < 0$ and $t_{f_i}^{\psi_1} - t_{r_j}^{\psi_2} > 0$ since the minimum width of overlapping interval of ψ_1 and ψ_2 is greater than 0.

If $t_{f_i}^{\psi_2} < t_{f_i}^{\psi_1}$, it yields

$$t_{r_j}^{\psi_2} - b < t_{f_i}^{\psi_2} - a$$

or

$$t_{f_i}^{\psi_2} - t_{r_j}^{\psi_2} > a - b$$

which is true since the width positive interval of ψ_2 is greater than 0 and $a-b < 0$.

Part ii): We need to show $t_r^{\mathcal{U}}$ is the first instance of time that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is true. Assume $t_{r_2}^{\mathcal{U}}$ is the first time that ψ_2 is satisfied. So, $t_{r_2}^{\mathcal{U}} \in [t_r^{\mathcal{U}} + a, t_r^{\mathcal{U}} + b]$. As a result, $t_r^{\mathcal{U}} + a < t_{r_2}^{\mathcal{U}} < t_r^{\mathcal{U}} + b$. Since we look for the first time that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is satisfied, so:

$$t_r^{\mathcal{U}} > t_{r_2}^{\mathcal{U}} - b \tag{B.1}$$

Besides, ψ_1 must be true from $t_r^{\mathcal{U}}$ until $t_{r_2}^{\mathcal{U}}$. So, we have:

$$t_r^{\mathcal{U}} > t_{r_1}^{\mathcal{U}} \tag{B.2}$$

$$t_{r_2}^{\mathcal{U}} > t_{f_1}^{\mathcal{U}} \tag{B.3}$$

Condition B.3 is always true since the minimum overlapping interval of ψ_1 and ψ_2 is greater that 0. Combining B.1 and B.2, one can compute the first time (rising time) that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is satisfied as: $t_r^{\mathcal{U}} = max(t_{r_i}^{\psi_1}, t_{r_j}^{\psi_2} - b)$

Part iii): We need to show that $t_f^{\mathcal{U}}$ is the final time that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is true.

There are two cases that indicates a falling timestamp:

a) there does not exist a time $t' > t_{f_2}^{\mathcal{U}} \in [t_f^{\mathcal{U}} + a, t_f^{\mathcal{U}} + b]$ such that ψ_2 is true and

b) there exists a time $t' > t_{f_2}^{\mathcal{U}} \in [t_f^{\mathcal{U}} + a, t_f^{\mathcal{U}} + b]$ such that ψ_2 is true <u>but</u> there exists a time $t'' > t_{f_2}^{\mathcal{U}} \in [t_f^{\mathcal{U}} + a, t_f^{\mathcal{U}} + b]$ such that ψ_1 is not true.

For the first case, assume $t_{f_2}^{\mathcal{U}}$ is the final time that ψ_2 is satisfied. So, $t_{f_2}^{\mathcal{U}} \in [t_f^{\mathcal{U}} + a, t_f^{\mathcal{U}} + b]$.

As a result, $t_f^{\mathcal{U}} + a < t_{f_2}^{\mathcal{U}} < t_f^{\mathcal{U}} + b$. As we look for final time that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is satisfied, so:

$$t_f^{\mathcal{U}} < t_{f_2}^{\mathcal{U}} - a \tag{B.4}$$

In second case, ψ_1 should be true from $t_f^{\mathcal{U}}$ until $t_{f_2}^{\mathcal{U}}$. Then:

$$t_f^{\mathcal{U}} > t_{r_1}^{\mathcal{U}} \tag{B.5}$$

$$t_{f_1}^{\mathcal{U}} > t_{f_2}^{\mathcal{U}} \tag{B.6}$$

Since we are looking for the final time that ψ_2 is satisfied but ψ_1 is not, we can replace t_{f_2} with $t_f + a$. As a result,

$$t_f^{\mathcal{U}} > t_{f_1}^{\mathcal{U}} - a \tag{B.7}$$

Combining B.4 and B.7, one can compute the final time that $\psi_1 \mathcal{U}_{[a,b]} \psi_2$ is satisfied as $t_f^{\mathcal{U}} = min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$.

Condition B.5 is satisfied if we replace $t_f^{\mathcal{U}}$ with $min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a$.

So, we have $min(t_{f_i}^{\psi_1}, t_{f_i}^{\psi_2}) - a > t_{r_1}^{\mathcal{U}}$ which is always true because both the width of positive interval of ψ_1 and the minimum overlapping interval of ψ_1 and ψ_2 are greater than a. □

Lemma B.1.5. If the width of overlapping interval of ψ_1 and ψ_2 is less than a (a computed falling timestamp is less than previous computed rising one), we can cancel them out and $\psi_1\mathcal{U}_{[a,b]}\psi_2$ remains false in that interval ($[t^{\mathcal{U}}_{f_i}, t^{\mathcal{U}}_{r_i}]$).

Proof. According to Lemma B.1.4, $t^{\mathcal{U}}_{r_i}$ is the first time that $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is true. So, for all $t^{\mathcal{U}}_{f_{i-1}} < t < t^{\mathcal{U}}_{r_i}$, $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is false. Besides, $t^{\mathcal{U}}_{f_i}$ is the final time that $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is true. So, for all $t^{\mathcal{U}}_{f_i} < t < t^{\mathcal{U}}_{r_{i+1}}$, $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is false. As a result, for all times $t \in [t^{\mathcal{U}}_{f_i}, t^{\mathcal{U}}_{r_i}]$, $\psi_1\mathcal{U}_{[a,b]}\psi_2$ is false. □

B.2 The TTL Rules

Logical systems in Mathematical Logic are sound if and only if every formula that can be proved in the system is logically valid with respect to the semantics of the system. In order to have the Soundness proof for TTL, it is firstly needed to have some proofs for the TTL rules. In this section, it is shown that all other timing specifications can be converted to Latency constraint, and then the proof for the TTL Soundness is represented.

In chapter 4, there are 7 rules for TTL. The proof for LE is in Theorem 4.3.1. The rest of the proofs are in this section.

B.2.1 Simultaneity

Theorem B.2.1. *The rule SL is valid. The simultaneity timing constraint is represented in terms of latency constraints. If two or more events occur simultaneously within ε time units, the latency between any two events (mentioned in the simultaneity constraint) is less than ε.*

$\vdash \mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon) \rightarrow \vdash \mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon, \quad where \quad i, j \in \{1, 2, ..., n\},\ i < j$

Proof. Consider $A : \mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$ and $B : \mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon$. For A, if $\nvdash \mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$, regardless of expression B, the entire formula is $\top$. Therefore, if A is $\top$, B should be $\top$. Hence, we assume $\vdash \mathcal{S}(\phi_1, \phi_2, ..., \phi_n, \varepsilon)$. If $\nvdash \mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon$, we have $(\varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)) \vee (\mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon)$. Assume $C : \varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)$ and $D : \mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon$.

i) For term $C : \varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)$ we have $(\sigma, t) \models \Big((0 < \tau(\phi_j) - \tau(\phi_i)) \wedge (\frac{0-\varepsilon}{\delta} + 2 < \tau(\phi_j) - \tau(\phi_i))\Big)$ iff $(\sigma, t) \models 0 < \tau(\psi_j) - \tau(\phi_i)$ and $(\sigma, t) \models \frac{\varepsilon - 0}{\delta} + 1 < \tau(\phi_j) - \tau(\phi_i)$. Considering $\exists t' \in \mathbb{N}, t' = \tau(\phi_1)$ and $\exists t'' \in \mathbb{N}, t'' > t'$ s.t. $t'' = \tau(\phi_2)$ and $(s, t'') \models \phi_2$. We have $\frac{\varepsilon}{\delta} + 1 < t'' - t'$. Assume $E : \frac{\varepsilon}{\delta} + 1 < t'' - t'$.

From term A, we know: $\forall t_i \in \mathbf{N}, i \in \{1, 2, ..., n\}$ s.t. $(\sigma, t_i) \models \phi_i$ and $max\{\tau(\phi_i)\} - min\{\tau(\phi_i)\} < \varepsilon$. Since max and min are chosen, any pair of $\{\phi_p, \phi_q\}; p, q \in \{1, 2, ..., n\}$, and $p < q$ satisfy term A. We choose ϕ_i and ϕ_j where $i < j$. Hence, $(\sigma, t_i) \models \phi_i$, $(s, t_j) \models \phi_j$, and $\tau(\phi_j) - \tau(\phi_i) < \frac{\varepsilon}{\delta} - 1$. Thus, $t'' - t' < \frac{\varepsilon}{\delta} - 1$. If we use $A5$ in table 4.1, it becomes $1 - \frac{\varepsilon}{\delta} < t' - t''$ and we call it F. By applying axiom $A4$ in table 4.1 on terms E and F and we have $2 < 0$ which is not possible. Therefore, we have contradiction and our assumption ($\nvdash \mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon$) is not true.

ii) For term $D : \mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon$, from *Theorem* 4.3.1 we know that $(\mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon) \rightarrow (\varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)) \wedge (\mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon)$. Since based on last proof, $\varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)$ is not valid, and there is a conjunction in term D, $(\mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon)$ also is not true. □

B.2.2 Chronological

Theorem B.2.2. *The CL rule is valid. If the occurrence of two event is more than ε, they are chronological.*

$$\vdash \mathcal{C}(\phi_1, \phi_2, ...\phi_n, \varepsilon) \rightarrow \vdash \varepsilon < \mathcal{L}(\phi_i, \phi_{i+1}, 0), 1 \leq i < n-1$$

.

Proof. Consider $A : \mathcal{C}(\phi_1, \phi_2, ...\phi_n, \varepsilon)$ and $B : \varepsilon < \mathcal{L}(\phi_i, \phi_{i+1}, 0)$

In B part, we consider $\nvdash \varepsilon < \mathcal{L}(\phi_i, \phi_{i+1}, 0)$. Hence, we have:

$(\mathcal{L}(\phi_i, \phi_{i+1}, 0) < \varepsilon) \vee (\mathcal{L}(\phi_i, \phi_{i+1}, 0) = \varepsilon)$

Assume $C : \mathcal{L}(\phi_i, \phi_{i+1}, 0) < \varepsilon$ and $D : \mathcal{L}(\phi_i, \phi_{i+1}, 0) = \varepsilon$

For C: $(\sigma, t) \models \Big((0 < \tau(\phi_{i+1}) - \tau(\phi_i)) \wedge (\tau(\phi_{i+1}) - \tau(\phi_i) < \varepsilon)\Big)$

iff $(\sigma, t) \models 0 < \tau(\psi_{i+1}) - \tau(\phi_i)$ and $(\sigma, t) \models \tau(\phi_{i+1}) - \tau(\phi_i) < \varepsilon$

iff $0 < \tau(\phi_{i+1}) - \tau(\phi_i)$ and $\tau(\phi_{i+1}) - \tau(\phi_i) < \varepsilon$

Assume $E : \tau(\phi_{i+1}) - \tau(\phi_i) < \varepsilon$

form A, we know: $\forall t_i \in \mathbf{N}, i \in \{1, 2, ..., n\}$, $(\sigma, t_i) \models \phi_i$ and $(\sigma, t_{i+1}) \models \phi_{i+1}$ s.t. $t_i + \varepsilon < t_{i+1}$.

Therefore, $\varepsilon < \tau(\phi_{i+1}) - \tau(\phi_i)$ and it contradicts with E.

For D, we know $D = \mathcal{L}(\phi_i, \phi_j, 0) = \varepsilon) \rightarrow (\varepsilon < \mathcal{L}(\phi_i, \phi_j, 0)) \wedge (\mathcal{L}(\phi_i, \phi_j, 0) < \varepsilon)$

form the last part we know C is not satisfied, thus, it is not true too.

□

B.2.3 Frequency

Before having the proofs for the Frequency and Phase axiom/rules, we need to have the definition for the sequence of events, the definition for ϕ^k and ϕ^{k-1}.

Definition B.2.1. ϕ^{k-1} and ϕ^k are two consecutive events on the same signal σ.

$$\exists t, t' \in \mathbb{N}, \quad s.t. \quad (\sigma, t) \models \phi \quad and \quad (\sigma, t') \models \phi \quad \nexists t'' \in \mathbb{N} \quad s.t. \quad t < t'' < t', (\sigma, t'') \models \phi$$

Theorem B.2.3. *The FL rule is valid. If the frequency of an event on the signal σ is less than a certain frequency, the latency between each two consecutive events on σ are greater than a certain value.*

$$\mathcal{F}(\phi, \varepsilon) < f \rightarrow \frac{1}{f+\varepsilon} < \mathcal{L}(\phi^{k-1}, \phi^k, 0)$$

Proof. Consider $A : \mathcal{F}(\phi, \varepsilon) < f$ and $B : \frac{1}{f+\varepsilon} < \mathcal{L}(\phi^{k-1}, \phi^k, 0)$ where $\varepsilon < f$

In B statement, we assume $\nvdash \frac{1}{f+\varepsilon} < \mathcal{L}(\phi^{k-1}, \phi^k, 0)$, hence, we have:

$\frac{1}{f+\varepsilon} < \mathcal{L}(\phi^{k-1}, \phi^k, 0) \vee \mathcal{L}(\phi^{k-1}, \phi^k, 0) = \frac{1}{f+\varepsilon}$.

considering $C : \mathcal{L}(\phi^{k-1}, \phi^k, 0) < \frac{1}{f+\varepsilon}$ and $D : \mathcal{L}(\phi^{k-1}, \phi^k, 0) = \frac{1}{f+\varepsilon}$ for C statement, we have:

$\tau(\phi^k) - \tau(\phi^{k-1}) < \frac{1}{f+c}$ and $0 < \tau(\phi^k) - \tau(\phi^{k-1})$.

Considering $t' = \tau(\phi^{k-1}), t'' = \tau(\phi^k)$

Therefore, by $(\sigma, t') \models \phi$ and $t'' > t'$ s.t. $(\sigma, t'') \models \phi$ we have $E : t'' - t' < \frac{1}{f+c}$.

From A we know that $\exists t' \in \mathbf{N}$ s.t. $(\sigma, t') \models \phi$, $\exists t'' > t'$ s.t. $(\sigma, t'') \models \phi$, $\nexists t''', t' < t''' < t''$ s.t. $(\sigma, t''') \models \phi$, and $\frac{1}{f+\varepsilon} < t'' - t'$ which contradicts with E [1]

.

□

Theorem B.2.4. *The FE rule is valid. The Exact Frequency equals the conjunction of Maximum and Minimum Frequency constraints.*

$$\vdash \mathcal{F}(\phi, \varepsilon_f) = f \vdash \Big(f < \mathcal{F}(\phi, \varepsilon_f) \wedge \mathcal{F}(\phi, \varepsilon_f) < f\Big)$$

Proof. Consider $A : \mathcal{F}(\phi, \varepsilon_f) = f$, $B : f < \mathcal{F}(\phi, \varepsilon_f)$, and $C : \mathcal{F}(\phi, \varepsilon_f) < f$.

For term A, if $\nvdash \mathcal{F}(\phi, \varepsilon_f) = f$, regardless of B and C, the entire formula is $\top$. Hence, we suppose A is $\top$ so we need to prove $B \wedge C = \top$.

For B, $0 < \tau(\phi^k) - \tau(\phi^{k-1})$ and $\tau(\phi^k) - \tau(\phi^{k-1}) < \frac{l-\varepsilon_f}{\delta} - 1$. For C, $0 < \tau(\phi^k) - \tau(\phi^{k-1})$ and $\frac{f+\varepsilon}{\delta} + 1 < \tau(\phi^k) - \tau(\phi^{k-1})$. Assume, $t' = \tau(\phi^{k-1})$ and $t'' = \tau(\phi^k)$, $\frac{f+\varepsilon}{\delta} + 1 < t'' - t' < \frac{f-\varepsilon}{\delta} - 1$ which is the same as the semantic definition for exact latency in Table 3.1, line 11.

□

B.2.4 Phase

Theorem B.2.5. *The PL rule is valid. If the Phase events on two different signals σ_1 and σ_2 are less than a certain value (p), the latency between the corresponding event is less than p.*

$$\vdash \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p \rightarrow \vdash \Big(\mathcal{F}(\phi_1, \varepsilon_f) = \mathcal{F}(\phi_2, \varepsilon_f)\Big) \wedge \mathcal{L}(\phi_1, \phi_2, \varepsilon_p) < p$$

Proof. Assume $A : \mathcal{F}(\phi_1, \varepsilon_f) = \mathcal{F}(\phi_2, \varepsilon_f)$ and $B : \mathcal{L}(\phi_1^k, \phi_2^k, \varepsilon_p) < p, \forall k \in \mathbb{N}, 1 \leq k \leq n$.

We consider the left part ($\mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$) is $\top$. Hence, we should show the right part is $\top$ as well. Therefore, from PXE in table 4.2, we know that the the two frequencies are identical (A is $\top$). Now, it is enough to show B is $\top$.

Consider B is not $\top$. Thereby, $\nvdash \mathcal{L}(\phi_1^k, \phi_2^k, \varepsilon_p) < p$. Since the frequency of the events on two signals σ_1 and σ_2 are the same, we take just one iteration of the events. We take the iteration number k. Then we can generalize the proof for the rest of the iterations.

Assume $t' = \tau(\phi_1^k)$ and $t'' = \tau(\phi_2^k)$ in $p < \mathcal{L}(\phi_1^k, \phi_2^k, \varepsilon_p)$. Therefore, $(\sigma, t') \models \phi_1$ and $(\sigma, t'') \models \phi_2$ and $\frac{p-\varepsilon_p}{\delta} < t'' - t'$. However, it contradicts the semantics in table 3.1,

[1]The proof for the relationship between Minimum Frequency and Maximum Latency is very similar to this proof.

line 13 where the maximum phase semantic is expressed. Therefore, assuming $\nvdash \mathcal{L}(\phi_1^k, \phi_2^k, \varepsilon_p) < p$ is not right.

□

Theorem B.2.6. *The PE rule is valid. The Exact Phase equals to the conjunction of Maximum and Minimum Phase constraints.*

$$\vdash \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p \vdash \Big(p < \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) \wedge \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p\Big)$$

Proof. Consider $A : \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p$, $B : p < \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p)$, and $C : \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) < p$.

From A, we know that $\mathcal{F}(\phi_1, \varepsilon_f) = \mathcal{F}(\phi_2, \varepsilon_f)$,$0 < \tau(\phi_1^k) - \tau(\phi_1^{k-1})$, and $0 < \tau(\phi_2^k) - \tau(\phi_2^{k-1})$. For term A, if $\nvdash \mathcal{P}(\phi_1, \phi_2, \varepsilon_f, \varepsilon_p) = p$, regardless of B and C, the entire formula is $\top$. Hence, we suppose A is $\top$ so we need to prove $B \wedge C = \top$.

For B, $0 < \tau(\phi_1^k) - \tau(\phi_1^{k-1})$, $0 < \tau(\phi_2^k) - \tau(\phi_2^{k-1})$, and $\frac{p-\varepsilon_p}{\delta} + 1 < |\tau(\phi_1^k) - \tau(\phi_2^k)|$.

For C, $0 < \tau(\phi_1^k) - \tau(\phi_1^{k-1})$, $0 < \tau(\phi_2^k) - \tau(\phi_2^{k-1})$, and $|\tau(\phi_1^k) - \tau(\phi_2^k)| < \frac{p+\varepsilon_p}{\delta} - 1 <$.

Assume, $t' = \tau(\phi_1^k)$ and $t'' = \tau(\phi_2^k)$, $\frac{p-\varepsilon_p}{\delta} + 1 < |t'' - t'| < \frac{p+\varepsilon_p}{\delta} - 1$ which is the same as the semantic definition for exact latency in Table 3.1, line 14.

□

B.2.5 TTL Soundness

Theorem B.2.7. *TTL is Sound. The language $TTL_\chi(V)$ is Sound. Let* $\mathbf{A}$ *be a formula and* $\mathbf{F}$ *a set of formulas in* **TTL**. *If* $\mathbf{F} \vdash \mathbf{A}$ *then* $\mathbf{F} \models \mathbf{A}$. *In particular, if* $\vdash \mathbf{A}$ *then* $\models \mathbf{A}$.

Proof. The proof runs by induction on the assume derivation of $\mathbf{A}$ from $\mathbf{F}$.

i) $\mathbf{A}$ is an axiom of $TTL_\chi(V)$. All axioms of TTL are valid and they can be proven by their semantics (see Theorem 4.3.2.

ii) Since $A \in F$, $F \models A$ is held trivially.

iii) All the rules are valid. The proofs are in Theorems B.2.1, Theorem B.2.2, Theorem B.2.3, Theorem B.2.5, Theorem B.2.6, Theorem 4.3.1, Theorem 4.3.2, and Lemma 4.4.1.

□