ABSTRACT

Title of dissertation: SEMANTIC-DRIVEN MODELING AND REASONING FOR ENHANCED SAFETY OF CYBER-PHYSICAL SYSTEMS

Leonard Petnga, Doctor of Philosophy, 2016

Dissertation directed by: Doctor Mark Austin
Associate Professor, Department of Civil and Environmental Engineering, and Institute for Systems Research

This dissertation is concerned with the development of new methodologies and semantics for model-based systems engineering (MBSE) procedures for the behavior modeling of cyber-physical systems (CPS). Our main interest is to enhance system-level safety through effective reasoning capabilities embedded in procedures for CPS design. This class of systems is defined by a tight integration of software and physical processes, the need to satisfy stringent constraints on performance, safety and a reliance on automation for the management of system functionality. Our approach employs semantic–driven modeling and reasoning : (1) for the design of cyber that can understand the physical world and reason with physical quantities, time and space, (2) to improve synthesis of component-based CPS architectures, and (3) to prevent under-specification of system requirements (the main cause of safety failures in software). We investigate and understand metadomains, especially temporal and spatial theories, and the role ontologies play in deriving formal, precise

models of CPS. Description logic-based semantics and metadomain ontologies for reasoning in CPS and an integrated approach to unify the semantic foundations for decision making in CPS are covered. The research agenda is driven by Civil Systems design and operation applications, especially the dilemma zone problem.

Semantic models of time and space supported respectively by Allen's Temporal Interval Calculus (ATIC) and Region Connectedness Calculus (RCC-8) are developed and demonstrated thanks to the capabilities of Semantic Web technologies. A modular, flexible, and reusable reasoning-enabled semantic-based platform for safety-critical CPS modeling and analysis is developed and demonstrated. The platform employs formal representations of domains (cyber, physical) and metadomains (temporal and spatial) entities using decidable web ontology language (OWL) formalisms. Decidable fragments of temporal and spatial calculus are found to play a central role in the development of spatio-temporal algorithms to assure system safety. They rely on formalized safety metrics developed in the context of cyber-physical transportation systems and collision avoidance for autonomous systems. The platform components are integrated together with Whistle, a small scripting language (under development) able to process complex datatypes including physical quantities and units. The language also enables the simulation, visualization and analysis of safety tubes for collision prediction and prevention at signalized and non-signalized traffic intersections.

Keywords: semantic, model-based systems engineering, description logics, ontology, time, space, cyber-physical system.

SEMANTIC-DRIVEN MODELING AND REASONING FOR ENHANCED SAFETY OF CYBER-PHYSICAL SYSTEMS

by

Leonard Petnga

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2016

Advisory Committee:
Associate Professor Mark Austin, Chair/Advisor
Professor John Baras
Professor Bilal M. Ayyub
Professor Paul Schonfeld
Assistant Professor Huan Xu

Acknowledgments

My graduate experience has been one that I will cherish forever, and I owe my thanks to all the people who have made this dissertation possible. First, I would like to express my sincere gratitude to my advisor Associate Professor Mark Austin for supporting my Ph.D studies and related research, for his patience, motivation, and immense knowledge. He has always made himself available for help and advice and has always kept his door open to me. Without his extraordinary theoretical ideas and computational expertise, this thesis would be a distant dream. Also, I would like to thank Professor John Baras for giving me an invaluable opportunity to work on challenging and extremely interesting projects in his laboratory. He has always provided constructive feedback and ideas, and multi-form assistance and advice that have contributed to the quality of this work while making my humble person a better researcher. Thank you to Professors Bilal M. Ayyub, Paul Schonfeld, and Elise Miller-Hooks, and Assistant Professor Huan Xu for serving on my thesis committee and reviewing the manuscript.

This thesis would have not been possible without financial sponsorship of the National Institute of Standards and Technology (NIST), and technical exchanges and discussions with its researchers, especially within the Systems Integration Division. Special thanks to Dr Vijay Srinivasan, Allison Barnard Feeney, Conrad Bock, Frank Riddick, Tina Lee, Peter O. Denno and Carlitta Foster-Hayes. Short-term funding from the US Naval and Air Systems Command (NAVAIR) and MITRE have contributed to this work too. Therefore, I thank Dr Huan Xu and Mark Blackburn

for making these opportunities available.

My colleagues at the Systems Engineering and Integration laboratory (SEIL) have enriched my graduate life in many ways and deserve a special mention. Dr Shahan Yang provided guidance and help during the early stages of the research. Parastoo Delgoshaei has contributed with prototype versions of codes – throughout the struggle, she has been a very loyal, dependable and great friend. Yuchen Zhou and Jacob Moschler helped with ideas and software support. Analyses performed by Maria Coehlo provided valuable insights into implications of space ontological commitments on design trade-off. My interactions with Seksun Moryade, James Jones, Chalida U-tapao, David Daily, Mehdi Dadfarnia, Ben Abeye, Nefretiti Nassar, Ron (Carina) Wang and Mamadou Faye have been very fruitful.

I would also like to acknowledge help and support of staff members at the Institute for systems Research (ISR) and at the Department of Civil and Environmental Engineering (CEE). Kimberly Edwards, Alexis Jenkins, Heather Stewart, Kerri Poppler James and Carla Scarbor's administrative help is highly appreciated. Thank you to Gwen Flasinski for computer support.

My housemates at my various places of residence have been a crucial factor in the smooth completion of this dissertation. Special thanks and gratitude to Marcus Williams and Derek Veil for reviewing countless manuscripts and Sonia Donaldson for her friendship. I would also like to thank Diane, Henrietta, Chantelle, Candace, Yaa, Isabelle, Sonia, Dominique, Nelie & Christian, Serge, Valerie & Noel, Liz & Olivier, Rosine & Emmanuel, Paul, Youssoufa and Gustave for their kindness and

words of support.

I owe my deepest thanks to my family who have always stood by me, guided me, and pulled me through impossible odds at times. I particularly thank my parents, siblings, and extended family in Cameroon, USA, France, Senegal, Equatorial Guinea and Canada. Special thanks to Za'a Made, Lili, Ariane & Ange, Malili & Allen, Sandrine & Gaeten, Sosso & Jackson, Mami J. & Max, Solange & Jacques, Gaby, Bruno, Aurelie & Alain for always cheering and encouraging me even during the most difficult moments of this research. I cannot list all of your names here, but be assured you have a special place in my heart. Words cannot express the gratitude I owe each and everyone of you.

It is impossible to remember and name all, and I apologize to those I've inadvertently left out. Lastly, thank you all and thank God!

Table of Contents

List of Figures

List of Abbreviations

AI	Artificial Intelligence
ALC	Attribute Language Concepts
API	Application Programming Interface
ARQ	Automatic Repeat Query
ASDE-X	Airport Surface Detection Equipment
ATIC	Allen's Temporal Interval Calculus
BFO	Basic Formal Ontology
BWI	Baltimore-Washington International (airport)
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CEM	Composite Entity Model
CPS	Cyber-Physical Systems
CPS-KMoDS	Knowledge Modeling and Decision Support framework for CPS
CPTS	Cyber-Physical Transportation System
CYCORD	CYClic ORDder
DA	Dimensional Analysis
DAE	Differential Algebraic Equations
DAML	DARPA Agent Modeling Language
DARPA	Defense Advanced Research Projects Agency
DL	Description Logics
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DSKB	Domain-Specific Knowledge Base
DSO	Domain-Specific Ontologies
DZ	Dilemma Zone
EFTP	Extended Fuzzy-Timing Petri net
ET	Eastern Time
EEM	Extended Entity Model
FAA	Federal Aviation Administration
FEA	Finite Element Analysis
FOL	First Order Logic
FSM	Finite State Machine
FSTPA	Formal System Theoretic Process Analysis
GIS	Geographic Information System
GML	Geographical Markup Language
GUI	Graphical User Interfaces
HPSG	Head-Driven Phrase Structure Grammar
IM	Intersection Manager
ISO	International Organization of Standardization
IRI	Internationalized Resource identifiers
JFMI	Java-based Functional Mock-up Interface
JTS	Java Topology Suite
JUMP	Java-based Unified Mapping Platform
LAN	Local Area Network
MBSE	Model-Based Systems Engineering

MMTS	Metamodeling Technical Space
MoC	Models of Computation
MoE	Measure of Effectiveness
MPC	Model Predictive Control
MVC	Model- View-Controller
NMPC	Nonlinear Model Predictive Control
ODE	Ordinary Differential Equation
OGC	Open Geospatial Consortium
OIL	Ontology Inference Layer
OOD	Object Oriented Design
OSM	Open Street Map
OTS	Ontological Technical Space
OWL	Web Ontology Language
PM	Primitive Model
PHYSYS	PHYsical SYStem
QHP	Quantified Hybrid Programs
RACER	Renamed ABox and Concept Expression Reasoner
RCC	Region Connectedness Calculus
RDF	Resource Description Framework
RDFS	Resource Description Framework Schema
RDQL	RDF Query Language
ROW	Right Of Way
SQL	Structured Query Language
SPARQL	Simple Protocol and RDF Query Language
STPA	System Theoretic Process Analysis
STT	Spatio-Temporal Trajectory
SUMO	Suggested Upper Merged Ontology
SUV	Suburban Utility Vehicle
SysML	System Modeling Language
TE	Temporal Elements
TF	Temporal Features
TM	Temporal Measures
TP	Temporal Properties
TR	Temporal Relations
TS	Temporal Structures
UAV	Unmanned Aerial Vehicle
UML	Unified Modeling Language
U.S.	United States
V2I	Vehicle To Infrastructure
V2V	Vehicle To Vehicle
W3C	World Wide Web Consortium
WWW	World Wide Web
XML	Extensible Mark-up Language

Chapter 1: Introduction

1.1 Problem Statement and Contributions

Cyber-physical systems (CPS) are systems in which network of computational and physical elements are seamlessly integrated and tightly coupled. The general idea of cyber-physical systems is that ...

> **... embedded computers and networks will monitor and control the physical processes, usually with feedback loops where computation affects physical processes, and vice versa.**

Cyber-physical systems are now possible due to remarkable advances in sensing, computing, communications, and material technologies over the past few decades. The basic design requirement is that software and communications technologies will work together to deliver functionality that is correct and works with no errors. Looking ahead, not only is CPS is expected to find its way into a multitude of industries, from buildings (e.g., energy efficient buildings) to automotive (e.g., self-driving cars) through health care (e.g., smart heart implant) and manufacturing (e.g., self-organized production lines), but in many cases, CPS capabilities will allow for completely new kinds of engineering design [75, 194, 195, 218, 246, 285]. Figure

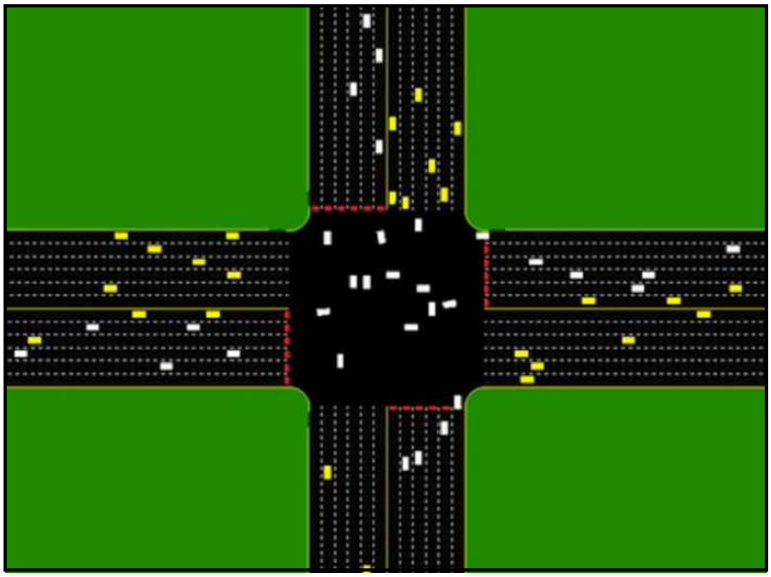


Figure 1.1: Behavior of self-driving automobiles at a busy traffic intersection – stop signs and traffic lights are replaced by mechanisms for vehicle-to-vehicle communication (Adapted from http:citylab.com)

1.1 shows, for example, the behavior of self-driving automobiles at a busy traffic intersection. Notice that the traffic lights are gone! Safety is achieved through the use of vehicle-to-vehicle communication, sensing (e.g., combinations of LiDAR, radar and GPS), and sophisticated algorithms and software for collision avoidance instead of stop signs and traffic lights.

Because CPS has the potential to fundamentally change the way in which we interact with the physical world [74, 204, 205], governmental entities and researchers have positioned it as the next technological revolution that will equal (and possibly surpass) the Internet. As this time, however, the realization of these opportunities is hindered by the lack of a foundational science and techniques for modeling CPS [191, 284].

1.1.1 Cyber-Physical System Components and Architectures

An examination of CPS application domains reveals components that span multiple physics and engineering domains, operate across multiple time scales, and have dynamics that are sometimes affected by human-in-the-loop interactions. Thus, we can categorize CPS components as follows [262]:

a) *Cyber components.* These are computation, control and communication platforms, each implementing some specific system function. Given their software (or cyber) nature, these components need a physical (or hardware) platform to run the corresponding program, to support communication among cyber components and with the surrounding environment.

b) *Physical components.* They act as facilitators for physical interactions as well as implementation of functional specifications for the system. Generally speaking, physical component complexity increases when components cover multiple engineering domains, and when components embed computational capability. Examples of the latter include on-board computers in automobiles, unmanned aerial vehicles (UAV), smart sensors in bridges, and smart medical implants.

Figure 1.2 shows the network structure and components in a prototypical CPS application. The system is made of four integrated and networked platforms with a physical plant. A network (wireless in this case) allows the various platforms to communicate with each others. This network could be as small as a Local Area Network(LAN) or as big as the Internet. Some of the links between the platforms are

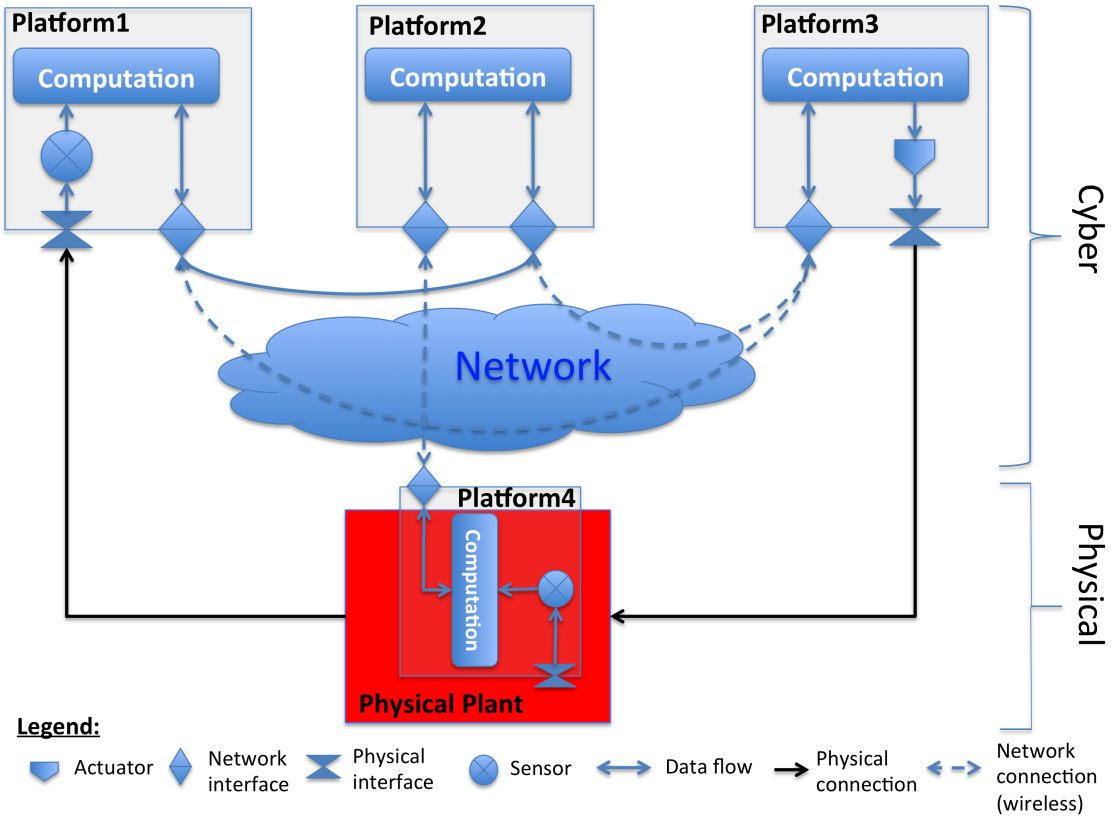


Figure 1.2: Schematic of a CPS (Adapted from [157]).

direct and would not go through the wireless network. One of the platforms (#4) is embedded in the physical plant which interacts with the cyber world through physical interfaces. Each platform is made of all or some of the following components.

i) *Computation module.* Computation modules process plant data collected by sensors and/or output from other platforms. System architectures may impose dependency relationships among computation modules, independently on their location. For our illustrative example (see Figure 1.2), this capability allows physical processes occurring in the plant to affect or modify computations in platform #2 using both the embedded platform (#4) and the wireless network

to communicate with platform #2.

ii) *Sensors*. Sensors collect plant data (physical measurements) and pass them to the computation module for further processing. For example, sensors are illustrated on platforms #1 and #4. They usually operate as a node in a sensor network architecture.

iii) *Actuators*. They intervene in the feedback control loop of the plant to control mechanisms or processes according to the system specifications. Platform #3 illustrates one of them.

iv) *Interfaces*. Network interfaces allow for the flow of data between platforms directly or through a network. Physical interfaces allow for plant and platform connectivity. In Figure 1.2, all platforms are equipped with both types of interfaces except for platform #2, which has only network interfaces.

1.1.2 Key Characteristics of CPS Component Interactions

The aforementioned description seems typical of modern software intensive systems. However, what sets CPS apart is the nature of the interaction between its components and their configuration. Especially, we can distinguish the following three characteristics.

1) *Two-way interactions between the cyber and physical subsystems*. As illustrated in Figure 1.2, CPSs go beyond sophisticated embedded systems with master-slave relationships to achieve cooperative, seamless, fully synergistic integra-

tion of the cyber and physical worlds [157, 189].

2) *Distributed system components.* CPS design deals with a multiplicity of physics and distributed components with concurrent behaviors [66, 158, 262]. The distribution of components can span a network as small as a local area network (LAN) or as big as the Internet. Some of the links between the platforms would not go through the wireless network.

3) *Embedded computational platforms.* With the current trend of increasing complexity of engineering systems, physical components could have some embedded computational capabilities. On-board computer in automobiles, Unmanned Aerial Vehicles (UAV), smart sensors in bridges and smart medical implants are some illustrations. This capability allows the effective feedback and communication between the physical and the cyber [189].

1.1.3 Contributions of this Dissertation

During the past four years the author has conducted research in ontological models and systems for safety-critical CPS and related problems. The scope of work has included: model-based design and formal verification processes for automated waterway system operations [213], ontological frameworks for knowledge modeling and decision support in cyber-physical systems [212], safe traffic intersections [211], connected-vehicle systems [206], security of unmanned aerial vehicles [214], spatial ontologies and models for safety-critical CPS [210], semantic platforms for CPS [209], ontologies of time and time-based reasoning [207].

The contributions of this thesis are documented in conference and journal publications [206–214], and can be summarized as follows:

Contribution 1: *Temporal and Spatial Semantics for Decidable Reasoning in CPS.* Procedures for semantic modeling of Time and Space are investigated and demonstrated. Allen's temporal interval and calculus were found to be the most qualified theory for the ontological description of this meatadomain in the context of CPS design. Similarly, the region connectedness calculus (RCC-8) has been qualified as spatial theory for CPS design. A corresponding compliant multi-scale spatial modeling was introduced and described.

Contribution 2: *Semantic-based Platform for Safety-critical CPS Modeling and Study.* A novel ontological-based knowledge and reasoning framework for decision support for CPS(CPS-KMoDS) was developed and described. The framework relies on OWL as ontological language and enables ontological description and integration of (application) domains, time and space as metadomains. Resulting models are determinate, executable and support physical quantities.

Contribution 3: *Safety Metrics, Algorithms and Analysis Methods for Safety-critical CPS.* We have developed, simulated and analyzed safety metrics that capture the essence of the interactions between entities as per safety theoretic analysis approaches. This was rendered possible thanks to lessons learned from dimension analysis. Types of collisions were investigated and corresponding avoidance strategies for away, glancing and clipping collisions were described. Collision avoidance algorithms, that effectively predict and resolve spatio-temporal conflicts in the cy-

ber world before they occur in the physical one, were developed and the impacts of spatial ontological commitment on decision making were investigated.

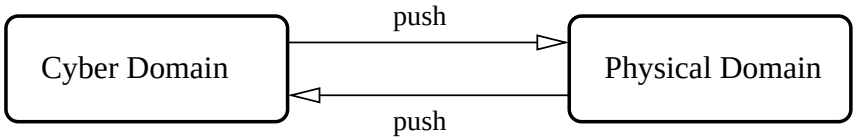
Contribution 4: *Java Library for spatio-temporal modeling and reasoning.* We have developed a library of software components for capturing and representing spatial and temporal knowledge and performing inference involving both metadomains. The library contains semantic (i.e., in the form of ontologies) and physical (i.e., for space) representations of the domains along with rules that enforce qualified theories in the respective domains. The open source Java Topology Suite (JTS) provides support in the form of high quality software for two-dimensional geometric representations of spatial entities.

1.2 Challenges in Cyber-Physical Design and Operation

1.2.1 Physical-Domain Behavior versus Cyber- Domain Behavior

Figure 1.3 provides a ten-to-twenty year perspective on cyber-physical systems design and operation.

A key challenge stems from the diversity of mathematical abstractions that are needed to describe behavior and failure across the physical and cyber domains. On the physical side of the problem, behavior tends to be continuous and, for the most part, can be expressed as the solution to ordinary and partial differential equations. Uncertainties can be managed through the use of reliability analysis and design safety factors. Usually, a physical system will provide some kind of warning



C−P Requirements

Behavior must be robust to unexpected conditions.
C−P system must be adaptable to sub−system level failures.

C−P Structure

Cyber capability in every physical component.
Executable code
Networks of computation
Heterogeneous implementations

Spatial and network abstractions
−− physical spaces
−− physical and social networks.
−− networks of networks
Sensors and actuators.

C−P Behavior

Dominated by logic
Control, communications
Stringent requirements on timing
Needs to be fault tolerant

Physics from multiple domains.
Combined logic and differential equations.
Not entirely predictable.
Multiple spatial− and temporal− resolutions.

Figure 1.3: Challenges in cyber-physical system design and operation.

– excessive displacements, cracking, heating, wear-out – if it is going to fail. Minor physical system behaviors are often localized. In contrast, the cyber side of the problem is dominated by computational systems that are discrete and inherently logical, with success tied to notions of correctness of functionality and timeliness of computation. If a computational strategy is logically incorrect, then "saying it louder" will not fix anything. Perhaps the most vexing aspect of computational systems design is that a small logical error can result in system-level failures that are very costly and, sometimes, even catastrophic. Solutions to this problem are complicated by the ease with which software development can begin before we have a full understanding of the system's purpose. As such, software-related accidents are usually caused by flawed requirements (and not standard wear-out failures),

erroneous assumptions about the operation of a control/computer system [171], and unsafe interactions among the system components and/or models of a process are inconsistent with the real state of the process and a controller provides unsafe control actions [163].

1.2.2 Safety of Cyber-Physical Systems

In order for a CPS to be safe, it must be able to adapt to both internal and environmental changes while maintaining data integrity and robustness. When system behaviors are uncertain and/or concurrent solutions to this problem are particularly vexing – the problem is so difficult that present-day modeling and design techniques are clearly inadequate [284]. A review of major accidents in modern engineering history highlights the shortfall of traditional safety analysis approaches and techniques to addressing safety in the design of modern engineering systems. Event chain models such as Heinrich's Domino Model [232] (or its Swiss Cheese Model [224] variant) are built on the premises that accidents are caused by direct, linear chains of events and often time point finger at human errors as partial [223] or sole cause [54] of accidents. Probabilistic risk models [200] were introduced to account for uncertainties that may arise in the chain of events with the side effect of explaining accidents as one in a billion occurrence assimilated to "simple coincidences" [175]. Despite being meticulously implemented, these state-of-the-art procedures have failed to stop or prevent catastrophes such as the loss of the Mars polar lander [136] or the Columbia space shuttle [249]. They also haven't

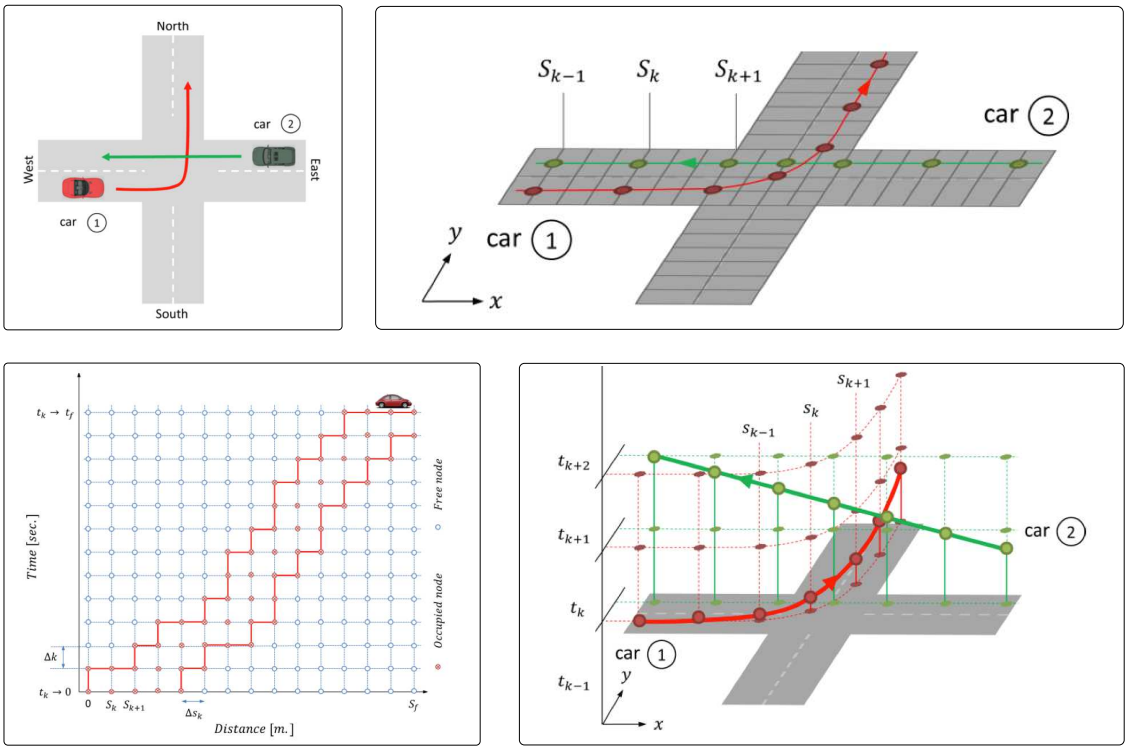


Figure 1.4: Collage of schematics for two cars that must safely traverse a traffic intersection (Adapted from [291]).

been able to prevent serious incidents such as the power-outage across Northeastern U.S. and Southeastern Canada in 2003 [181], or the emergency shutdown of the Hatch Nuclear Power Plant five years later, in 2008 [141]. Thus, safety researchers have been investigating approaches that account for the increasing central role software plays in managing system functionality and, unfortunately, in causing accidents [152, 161, 288]. Extensions of the McCalls software quality model [260] and system-theoretic models [160, 162] are such approaches.

In spatially distributed systems, safety challenges are often materialized in the form of risks of collision. Collision between dynamic entities is a permanent concern and has led researchers to investigate and develop strategies, algorithms

and systems to avoid collisions [122, 168, 266]. Figure 1.4 shows, for example, a series of schematics for two cars that need to safely traverse a traffic intersection. As indicated in the top left-hand schematic, one car wishes to go straight ahead; the second car wishes to make a left-hand turn. From a safely perspective, the key point to note is that the intersection space is a shared resource, which at all times can occupy at most one car. The scheduling of the car operations to avoid accidents can be viewed as the design of trajectories in space and time which must remain separated. See the lower right-hand schematic of Figure 1.4. Figure 1.5 illustrates the same ideas in the context of safe taxiing at airports.

1.3 State-of-the-Art Model-based Systems Engineering

The central tenet of model-based systems engineering (MBSE) is that systems should be designed and managed through the use of models [178, 193], as opposed to documents. MBSE procedures are driven by a need to achieve high levels of productivity in system development, and lead to design solutions that provide: (1) Bang for the buck – minimal mechanism; maximal function (i.e., a good, balance of functionality, performance and economics), (2) Reliable operation in a wide range of environments, and (3) Ease of accommodation for future technical improvements.

Established practice is to deal with design complexity through separation of concerns and development along disciplinary lines, followed by procedures for systems integration and validation and verification. While this approach eases work organization, design solutions tend to have loosely coupled system architectures that

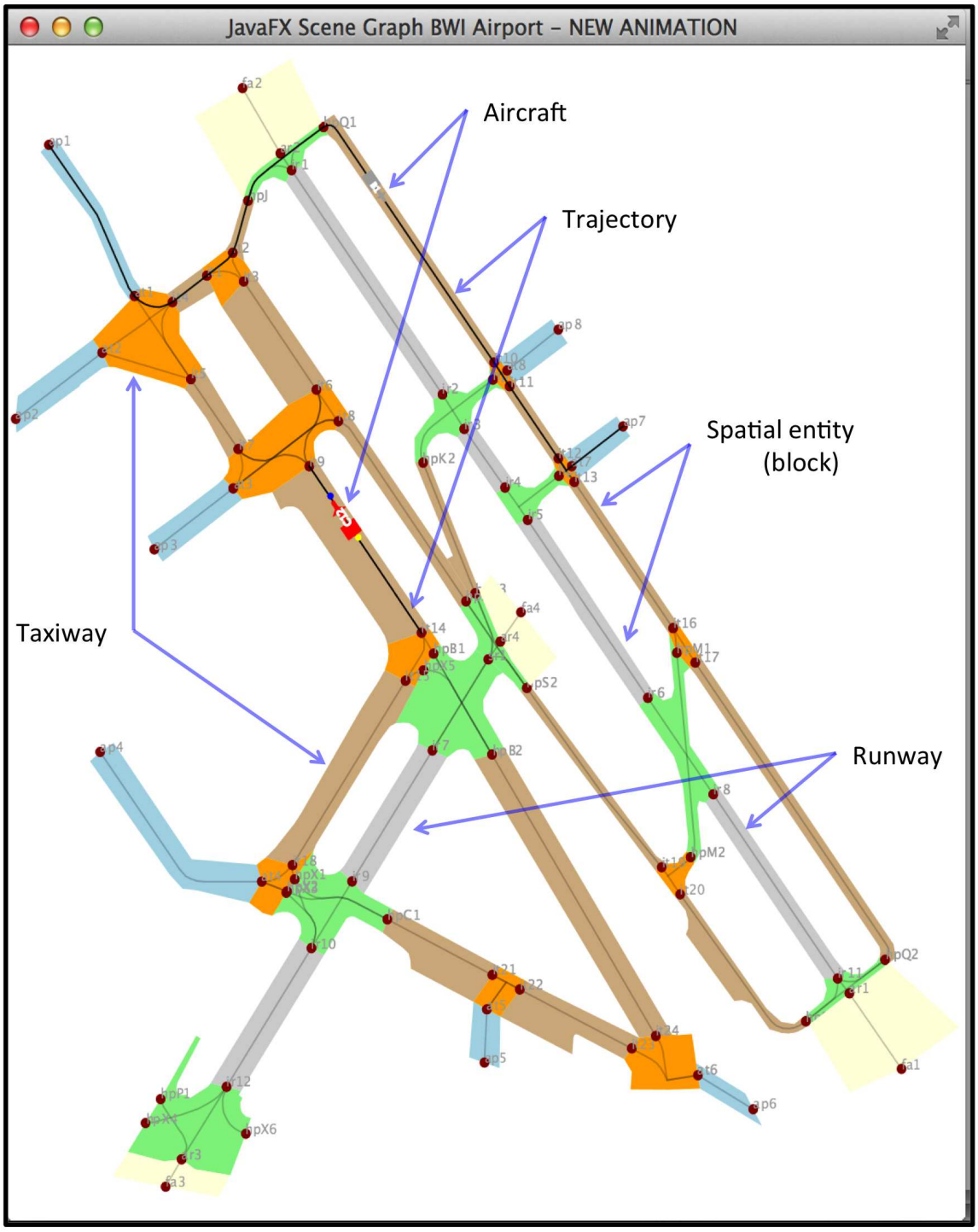


Figure 1.5: Airport Taxiway Modeling

are limited in levels of achievable performance. Increases in system size and complexity drive the need for: (1) disciplined approaches to system design that involve the application of decomposition, composition, abstraction and use of semi-formal and formal analysis [23, 25, 134, 184], and (2) modeling formalisms that capture cause-and-effect relationships between designer concerns (e.g., correctness of system functionality; adequacy of performance; assurance of safety) and problem solutions.

1.3.1 Multi-Level Approach Model-Based System Design

In a step toward addressing these concerns, Mosteller and co-workers [184] describe a multi-level approach to model-based system design having an intricate combination of mechanisms, as shown in Figure 1.6.

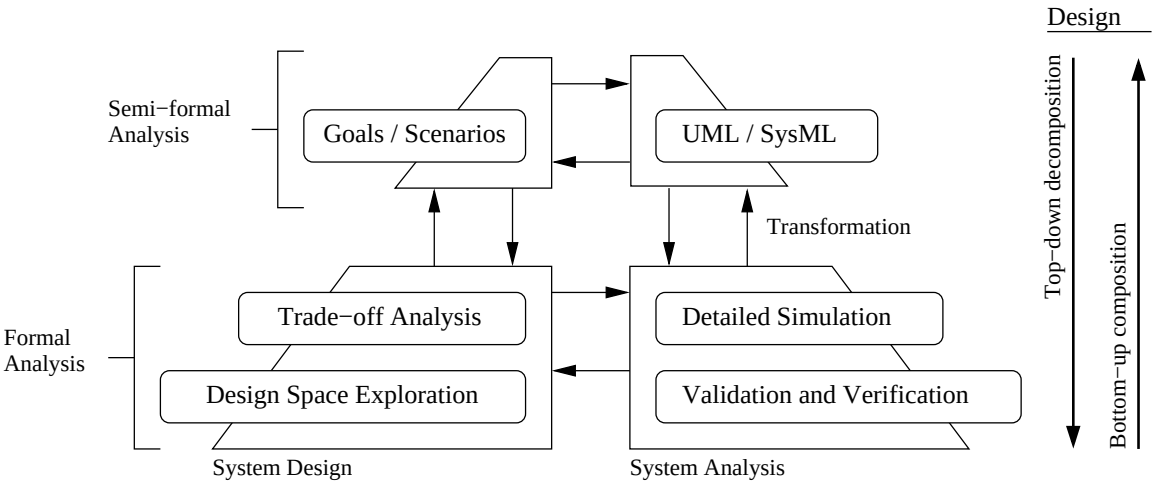


Figure 1.6: Multi-level approach model-based systems engineering. Semi-formal models provide a high-level view of the complete system (efficiency). Formal models provide a detailed view of the actual system (accuracy) [184].

The pyramid structure partitions the development effort into four interrelated blocks organized into two levels. The top level contains semi-formal models

capturing ideas (goal/scenarios) and preliminary designs represented in graphical languages such as the Unified Modeling Language (UML) and the System Modeling Language (SysML) [85, 271]. Together, goals and scenarios analysis and use of UML/SysML provide the designer with "big picture" summary of the system under development and highlight the major components, their connectivity, and performance. Representations for preliminary/tentative design need to be based on semi-formal models (e.g, UML and SysML) that have a fixed syntax and semantics and, thus, can be used to communicate ideas among the participating disciplines [85,271]. The lower level comprises models built from formal languages having precisely defined semantics. These models provide computational support for: (1) Detailed simulation of system behavior to assess achievable levels of performance, (2) Verification of correctness of functionality, particularly in the system control, and (3) Systematic design space exploration. Together the combination of high- and low-level representations work to prevent serious flaws in design direction, and to provide deep insight into the system behavior and functionality through formal analyses.

1.3.2 Pathway of System Development

Figure 1.7 shows the pathway from an operations concept to simplified models for behavior and structure, requirements, system-level design and model checking.

The first important task is to develop a functional description for what the system will do? Since a system does not actually exist at this point, these aspects of the problem description will be written as design requirements and mathematical

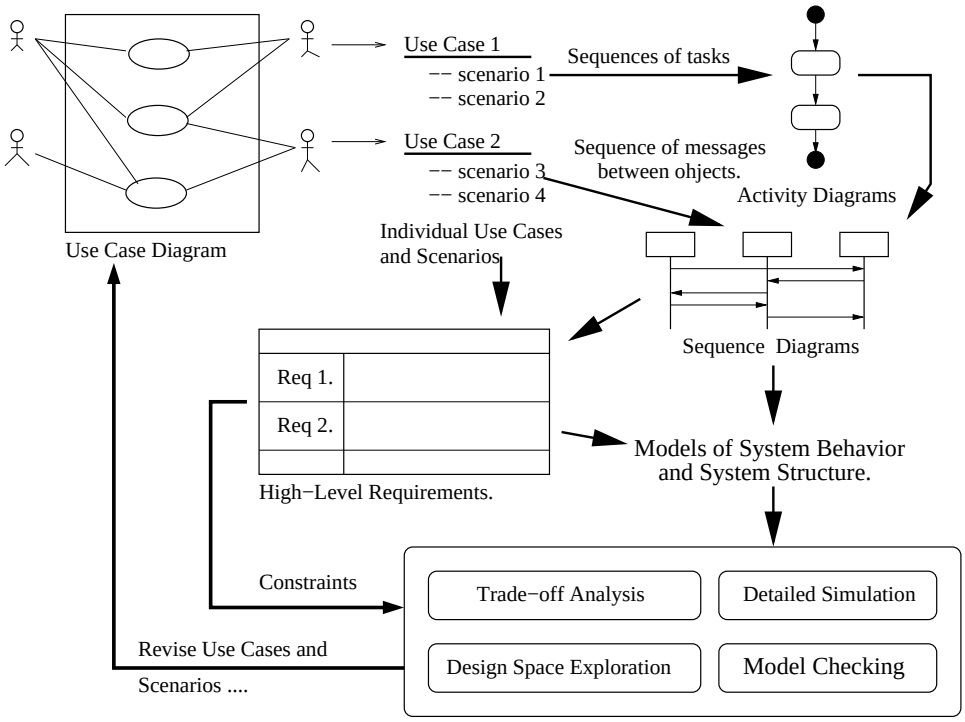


Figure 1.7: Pathway from operations concept to simplified models for behavior and structure, to requirements, system-level design and model checking.

constraints. It is important to note that while use cases and textual scenarios are neither requirements nor functional specifications, they imply requirements, objects, and object interactions and interfaces in the stories they tell. As a case in point, when use cases are associated with a specific class (in the system), working scenarios are, in essence, an invocation of the operations in the class. Some use cases will correspond to only a single operation. Others will involve a set of operations, usually occurring in a well-defined sequence. Further design requirements/constraints will be obtained from the structure and communication of objects in the models for system functionality (e.g., required system interfaces). Models of behavior specify what the system will actually do; often they can be represented as networks and hierarchies of tasks, functions and processes. Models of structure specify how the

system will accomplish its purpose. The system structure corresponds to collections of interconnected objects and subsystems, constrained by the environment within which the system must exist. The nature of each object/subsystem will be captured by its attributes, such as the physical structure of the design, environmental elements that will interact with the system, and the system inputs and system outputs. System-level design is created by mapping fragments of system functionality onto specific subsystems/objects in the system structure. Thus, the behavior-to-structure mapping defines in a symbolic way the functional responsibility of each subsystem/component. In the system evaluation, performance and characteristics of the system-level design are evaluated against the test requirements.

The heavy arrows in Figure 1.7 show pathways of traceability and iterations of refinement within the model-based development. Engineers should be able to look at a requirement and understand: (1) the goals and scenarios from which the requirements emanated, and (2) the ways in which the requirement has been satisfied in the system implementation. Pathways to requirement verification can involve a multitude of analytical procedures involving (continuous system) simulation for performance assessment, and formal approaches to analysis of (discrete) control actions for verification of correctness of system functionality. Usually, several iterations of development will be needed to modify the system behavior, system structure, perhaps even the original operations concept, and achieve a design that satisfies all of the system-level requirements.

1.4 Limitations of MBSE for CPS Design

1.4.1 Lack of Supportive Integration Science

Despite all of the advances that have been made in model-based systems engineering over the past few decades, the fact remains that today we do not have a mature science to support systems engineering of high-confidence cyber-physical systems assembled from subsystems spanning a multiplicity of domains. In order for cyber-physical design procedures to proceed in a rational way we need:

1. Mechanisms to easily combine abstractions from multiple physics (e.g., electrical, mechanical, chemical, biological) and field equations (solids, fluids, heat, electromagnetics, chemistry) into sets of coupled equations that model the system. Components may be discrete (as in rigid body elements, control actuation elements, software logic), or continuous.

2. Mechanisms for system assembly that will anticipate and deal with subsystem interactions, while also minimizing undesirable side effects and emergent behaviors. In other words, we ought to be able to compose CPS models from simpler well-defined systems.

3. A consistent treatment of time and space across multiple scales. This may leads to multiple models of the same field, which coexist in space and time. Examples of this class of problems occur in computational micro-mechanics and in fluid turbulence.

4. Mechanisms to understand how fault tolerance, security, decentralized control, and the social aspects of these systems influence design.

5. Methodologies and tools to conduct design-space exploration and trade-off analysis across domains that are part physical and part cyber.

Figure 1.8, adapted from Sztipanovits and co-workers [261, 262], summarizes the complexity and challenges in developing integrated architectures and models for CPS applications. Satisfying even a small subset of this vision is challenging. Lee [158] illustrates this complexity using a subset of an aircraft electrical power system (EPS). Depending on the domain-specific viewpoint, the perception of the system can range from a software to an electrical system passing by a mechanical, control or communication network. This leads to multiple domain-specific models of the CPS, with none of them covering the CPS entirely. In a slightly different take on strategies to address challenges for CPS development, Sztipanovits [262] explains this complexity through the observation that, often, the behavior of physical components in CPS is defined by interactions among multiple physics that are difficult to capture in a single model. Thus, the CPS designer will face the challenge of composition of multi-models for heterogeneous physical systems.

The integrated nature of CPS applications means that approaches to system development no longer work well. A second problem is due to the general trend toward software-dominated management of system functionality raises new concerns. For example, a new fundamental question is: How do we know that an automated driving system in a self-driving vehicle will always do the right thing? Present-

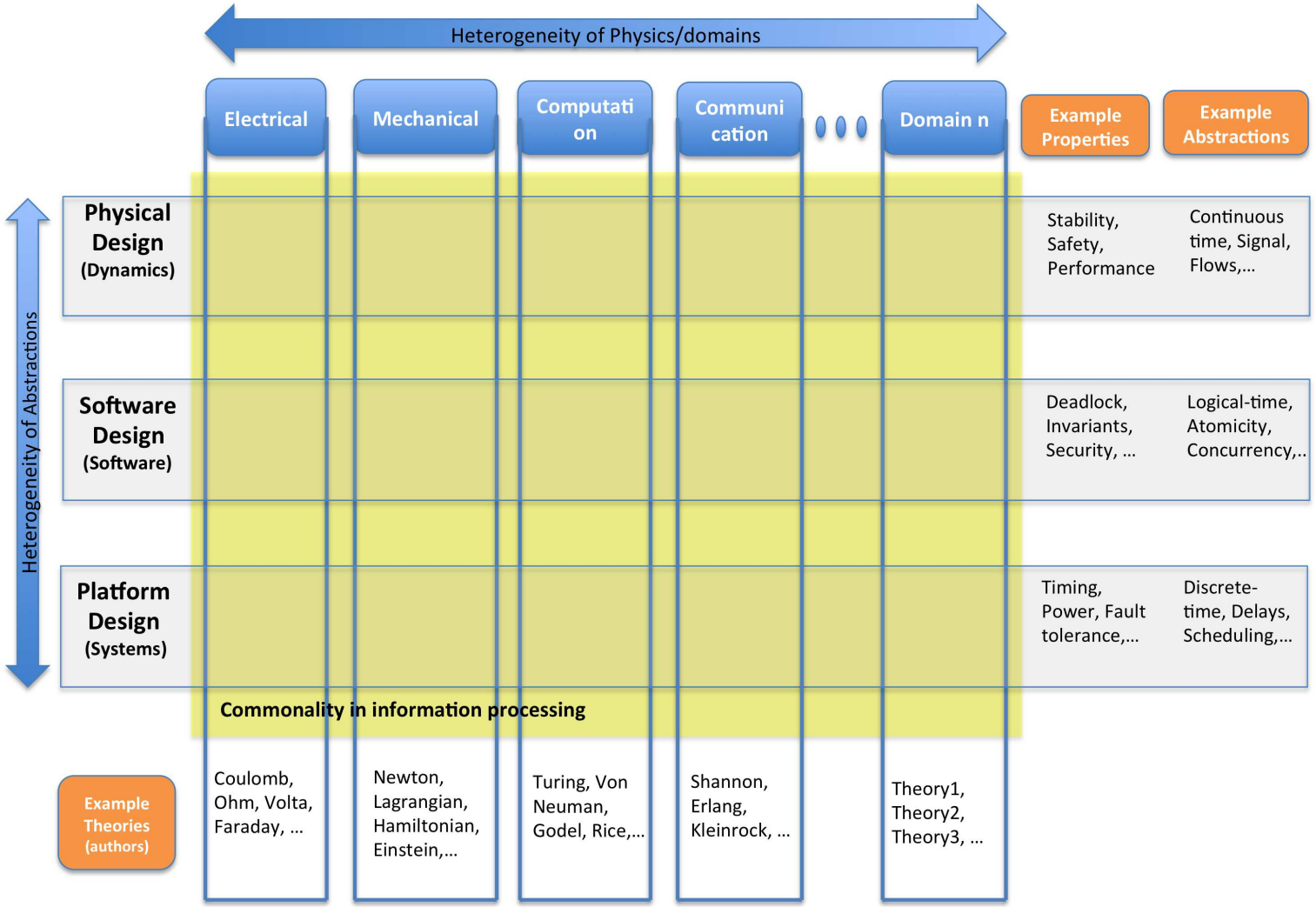


Figure 1.8: Complexity and challenges in CPS Modeling (Adapted from [262]).

day approaches to the model-based systems engineering and design of CPS lead to nondeterminate (i.e., broken) models with weak meta-domain (e.g. temporal, space) semantic support. This situation makes it difficult to analyze and evaluate critical system level behaviors and properties [154, 199, 295].

1.4.2 Deep but Fragmented Theories

With respect to CPS model semantics, Doyle [69] observes that theories backing the various disciplines involved in CPS are "deep but fragmented, incoherent and incomplete." The landscape of theories span from Turing and Von Neumann for computation to Einstein, Carnot or Newton for system physics through Nash and and Von Neumann for computation to Einstein, Carnot or Newton for system physics through Nash and Bode in control or Shannon in communication domain. Various domains involved in the modeling and design effort are orthogonally mapped to the main models abstraction layers. Unlike the software abstraction layer, platform and physical layers are obvious centers of attention for safety, mostly because of their physicality. However, a close look at deadlock properties in a software for CPS will lead us to consider it as a safety property for the system as well. The rationale here is that timing (from the physical world) in models at this abstraction layer is not a simple performance or quality factor for the software but a design correctness criterion. Therefore, answering the question on whether the system is safe or not at any point in time would require to consider all of its relevant aspects across the domains/physics involved and abstraction layers thus, various levels of

system integration. Unfortunately, at this time however, the lack of a "systems science" for system composition and integration, along with means to unify cyber and physical resources under the same mathematical framework remain the main obstacles toward the ultimate realization of the CPS vision [69, 192].

1.4.3 Limited Language and Domain Modeling Semantic Capabilities

As a solution to the problem of weak semantics of leading modeling languages - such as the Unified Modeling Language (UML) and System Modeling Language (SysML) - researchers have advocated approaches using logic to strengthen UML/SysML semantics through *language retrofitting*. Proponents of this approach advocate for the use logic to strengthen semantics of languages such as the UML [20, 34] or SysML [105]. This has made possible the development of computer aided support for automatic checking of model properties such as inconsistencies and redundancies during system design. In [106], the authors demonstrate the use of these language retrofitting approaches but also recognize their limitations when it comes to handling time. Moreover, the implementation of these approaches is not intuitive and needs strong knowledge in logic. The latter uses abstract mathematical notations thus, requires an extra effort from the designer. Other researchers have investigated the use of ontologies - especially OWL-based ones - to create formal language representations in system design environments (tools). System ontology as a pattern for what constitutes a system (parts, connections, identity, dependence, etc...) provides a strong foundation to the analyses needed for modeling a domain

of interest and establishing meaning of terms [104]. Thus, researchers have looked at the integration of ontological and system modeling languages mostly by means of *profiling* [103, 228, 277]. In the case of OWL and SysML, blocks and associations in the latter are mapped respectively to classes and properties in the former. Reasoning is performed by translating and interpreting model questions as axiom set questions.

1.5 Research Questions and Hypothesis

The separation of design concerns promoted by traditional systems engineering approaches, does not work for CPS. It leads to multiple distinctive viewpoints and a broken design flow that creates confusion and generate inconsistencies at every turn. Instead, the synergy between the physical (hardware) and cyber (software) subsystems for seamless integration becomes critical as the cyber (software) is increasingly responsible for the management of system functionality. Therefore, decisions made in the management of functionality have to be correct...all the time! To this end, the main research question we address is: how to improve the ability of the cyber to understand the physical world for efficient decision making?

Specifically, we would like to know:

1. *How to effectively identify, capture and express safety requirements and physical semantics in the overall CPS design flow?* One important motivation here is to uncover conditions, events or situations that could ultimately result into the system displaying emergent behaviors, setting it into a path toward unsafe

states. Another motivation is guaranteeing the decidability of the system reasoner in the context of high computational and expressiveness load, driven by the presence of complex data types (dimensions and units).

2. *What temporal and space theories are the most appropriate for modeling and design of CPS?* One major interest here is the ability to trace safety related questions to the interplay between space and time in spatio-temporal constraints and relationships that may or might be violated.

3. *What knowledge representation formalism is suitable for semantic-based modeling and reasoning in CPS?* Decidability, reasoning algorithms efficiency and support for concrete domains are of utmost importance in the context of CPS.

4. *To what extent can domain ontology models, especially the ones of time and space, and associated framework for formal reasoning about these meta-domains, be used to streamline design flows?* We ought to be able to obtain, from the resulting design flow, precise and accurate models that satisfy the requirements identified above and suitable for system level analyses.

5. *How can cyber and physical behaviors be seamlessly integrated into an executable CPS model ?* A successful attempt will lead to better models for simulation and analysis of CPS, which will provide greater insight into the design and understanding of such systems. Thus, the development of novel software infrastructure able to produce such models is critical to CPS systems engineers.

Our central hypothesis is that well-elicited requirements, use of formal languages and

proper capture of relevant domains semantics are the three main pillars to model and design correctness. This implies strong modeling language semantics and deep integration of domain semantics in models for analysis and formal verification of system requirements.

1.6 Research Scope and Objectives

The central tenet of our research is that CPS modeling and design challenges can be tackled through the development of ontological frameworks that embed of physical semantics into cyber models for system smartness. Accordingly, we wish to devise a platform infrastructure that will enable:

1. The identification, configuration and mapping of the appropriate instance of a semantic platform to the given system's engineering elements and components,

2. A cost-effective bottom-up composition of the system and multidisciplinary, multi-hierarchy and multi-domain traceability of cause-and-effect relationships and dependencies and,

3. The use of formal methods especially logic-based approaches to ensure the correctness of models of system functionality, system design and decision making.

If successful, this research will result in a framework that manages an interaction of system requirements with a variety of domains, and provides language support for CPS models having strong temporal, spatial, and domain-specific semantics.

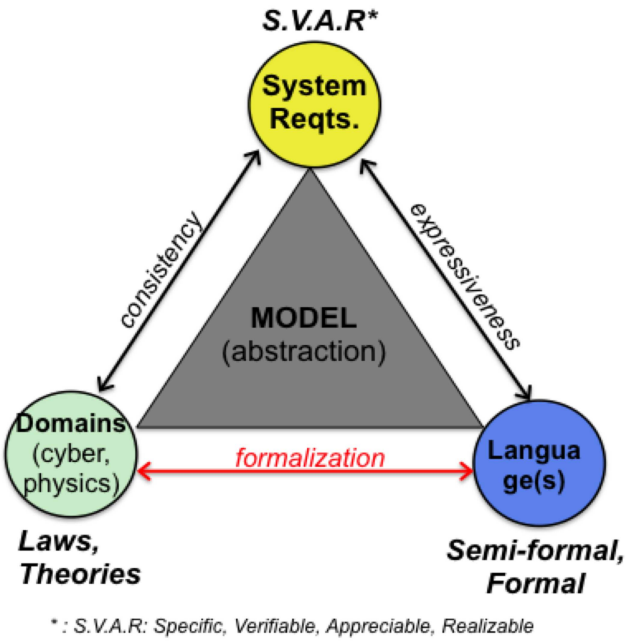


Figure 1.9: System requirements, domain, and language formalisms for integration and management of CPS models.

Figure 1.9 summarizes the coupling among these entities. The platform will be used to analyze and understand system-level performance and safety that depend on correct time- and space-based prediction of the future state of the system, and the satisfaction of physical world constraints that also depend on time and space.

The research will contribute to a computational infrastructure where meta-domain (i.e., time and space) and domain-specific ontologies, and rule checking routines operate hand-in-hand with a new scripting language called Whistle [65]. Figure 1.10 shows, for example, a vision for such an architecture. Engineering models of system structure will consist of networks and hierarchies of connected components formally described in terms of geometry (e.g., position, size) and connectivity (e.g., connected, touches, disjoint). Engineering models of system behavior will be combinations of discrete (e.g., statecharts) and continuous (e.g., differential equations)

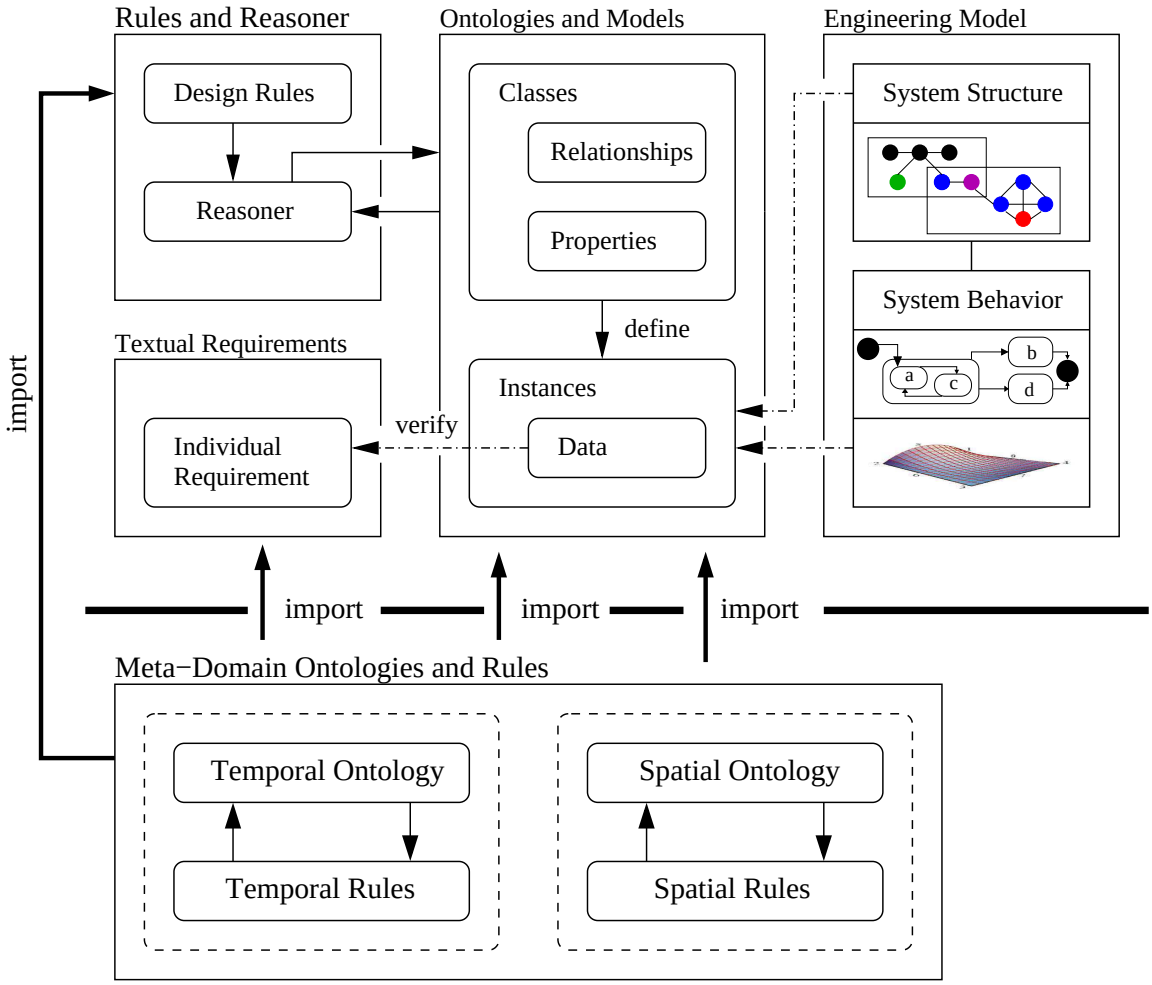


Figure 1.10: Framework for implementation of semantic-enabled simulation and rule-based control of cyber-physical systems. Domain-specific ontologies and rules are supported by meta-domain ontologies and rules covering time and space, which, in turn, are derived from theories and models of time and space described in Chapters 3 and 4 (Adapted from Delgoshaei, Austin and Pertzborn [65]).

behaviors.

The semantic counterpart of engineering models is ontologies (class hierarchies), individuals (graphs), and rules. An ontology as "a set of knowledge terms, including the vocabulary, the semantic interconnections, and some simple rules of inference and logic for some particular topic [117]." To provide a formal conceptualization within a particular domain, and thereby facilitate communication among people and machines, ontologies need to accomplish three things: (1) Provide a semantic representation of each entity and its relationships to other entities; (2) Provide constraints and rules that permit reasoning within the ontology, and (3) Describes behavior associated with stated or inferred facts. In the proposed architecture, ontologies and rules in the temporal and spatial domains will be integrated with domain-specific ontologies and rules, and support reasoning for simulation and rule-based control. Computation with rules provides several advantages [176, 229]: (1) Rules that represent policies are easily communicated and understood, (2) Rules retain a higher level of independence than logic embedded in systems, (3) Rules separate knowledge from its implementation logic, and (4) Rules can be changed without changing source code or underlying model. A rule-based approach to problem solving is particularly beneficial when the application logic is dynamic, and where rules are imposed on the system by external entities. Both of these conditions apply to the design and management of cyber-physical systems.

1.7 Dissertation Outline

This dissertation is organized into eight chapters. Chapter 2 introduces theories, languages and tools used in the Semantic Web. These tools and languages will be used extensively in our studies of time and space, and applications that can be built with these capabilities. Semantic models of time and space (and spatio-temporal combinations of space and time) are discussed in Chapters 3 and 4, respectively. Chapter 5 introduces a novel ontological framework for knowledge modeling and reasoning for CPS. Chapter 6 discusses the development and simulation of safety metrics for cyber-physical transportation systems design and analysis. Chapter 7 covers the formulation of spatio-temporal metrics and algorithms for collision avoidance in safety-critical CPS. And finally, the conclusions and contributions of this dissertation, and suggestions for future work are presented in Chapter 8.

Chapter 2: Semantic Web: Theory, Models, Languages, and Tools

2.1 Introduction to Semantic Web

2.1.1 Semantic Web Vision

In his conceptualization of the World Wide Web (late 1980s), Tim Berners-Lee [35] identified two main goals:

1. To make the Web a collaborative medium and,

2. To make the Web understandable and automatically processable by machines.

During the past twenty five years the first part of this vision has come to pass – today's Web provides a medium for presentation of data/content to humans. Machines are used primarily to retrieve and render information. Humans are expected to interpret and understand the meaning of the content. The Semantic Web aims to produce a semantic data structure which allows machines to access and share information, thus constituting a communication of knowledge between machines, and automated discovery of new knowledge [94, 117, 237]. Realization of this goal will require mechanisms (i.e., markup languages) that will enable the introduction, coordination, and sharing of the formal semantics of data, as well as an ability to rea-

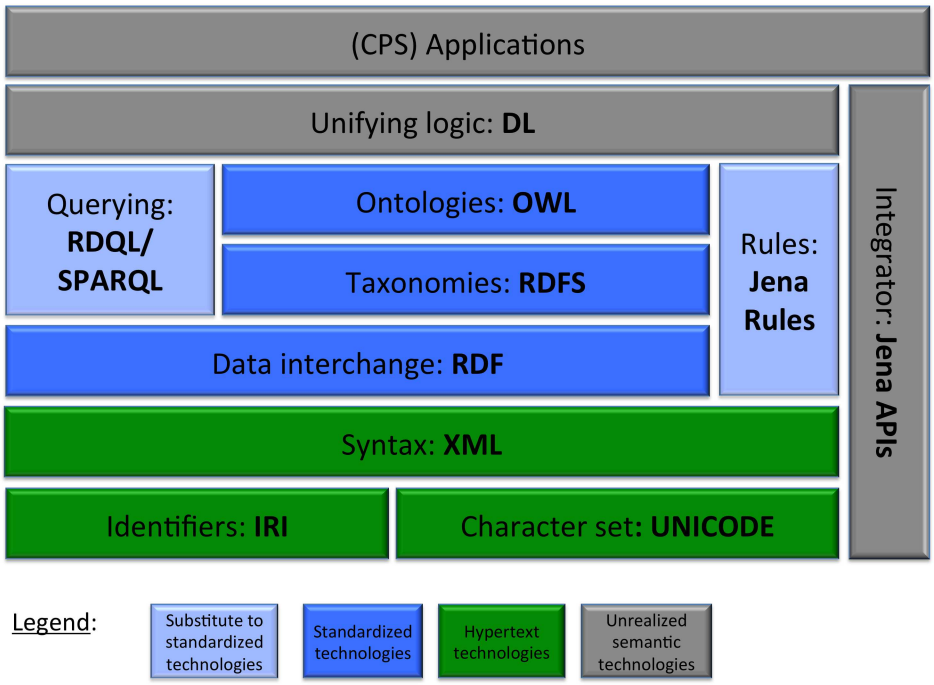


Figure 2.1: Semantic Web technology stack.

son and draw conclusions (i.e., inference) from semantic data obtained by following hyperlinks to definitions of problem domains (i.e., so-called ontologies).

2.1.2 Technical Infrastructure

Figure 2.1 illustrates the technical infrastructure that supports the Semantic Web vision, and the foundation upon which we hope to build CPS applications. Each new layer builds on the layers of technology below it. Briefly, the bottom layer is constructed of Internationalized Resource Identifiers (IRI) and Unicode. IRIs are a generalized mechanism for specifying a unique address for an item on the web. The eXtensible Markup Language (XML) provides the fundamental layer for representation and management of data on the Web. XML data is organized into tree

hierarchies. As already noted, Semantic Web applications will gather information from a variety of sources, and in the context of CPS, merge and organize these sources for decision making. Unfortunately, there is no easy way for tree structures to be merged. The resource description framework (RDF) solves this problem by allowing for the representation of graphs of data on the web. Graphs can always be merged. The web ontology language (OWL) provides for semantic descriptions of the underlying data. Together, XML, RDF and OWL allow for the implementation of reasoning that can prove whether or not assertions are true or false. For practical purposes, these tools need to operate in (almost) real time and, as such, description logics require extensions to make them computationally decidable.

2.2 Description Logics (Semantics and Ontologies for Reasoning)

2.2.1 Knowledge Representation Formalisms

Formal representation of knowledge of a domain requires formalisms that describe it. Thus, researchers have developed several knowledge representation formalisms such as Semantic Networks [17, 222, 256], Frame Systems [40, 76, 116, 183], Description Graphs [47, 203, 254] and Logic-based formalisms [27, 29] as illustrated in Figure 2.2. We briefly describe these formalisms as follows.

1. **Semantic Networks:** Information in this formalism is stored in categories that are logically related to each other in a hierarchy without repetition from one level to another. Multiple types of edges (subclass/superclass, prop-

erty/subproperty, and/or, etc..) and nodes (subject/object, generic/individual) offer ways to create a semantic network of the domain under study. Reading the graph allows one for instance to translate a subclass/superclass edge into a concept definition. This results in a label graph materializing a definitorial representation of (world) concepts. In [222], the author provides a full description of this formalism.

2. **Frame Systems:** This formalism has been introduced as a solution to the increasing complexity of semantic networks. Data structures similar to records are used to represent knowledge about situations and objects. Defaults, multiple perspectives and analogies are also included with the goal of regrouping all relevant knowledge about a situation in one object instead of having the information distributed across multiple axioms. In CLASSIC [257], frame-based formalisms are implemented in set of Lisp functions that turn out to be difficult in use for capturing arbitrary disjunctions.

3. **Description Graphs:** Graphs are given a formal semantics through a translation into first-order formulae as information representation formalism. Sowa [254, 255] introduces *conceptual graphs* as the basic and most important decidable fragment of this formalism. This allows its use in support to most reasoning services involving graphs validity or subsumption. The author views this formalism as a descendant of frame systems and semantic networks.

4. **Logic-based Formalisms:** They are an evolution of the declarative part of frame systems and they have played a central role in the evolution of artificial

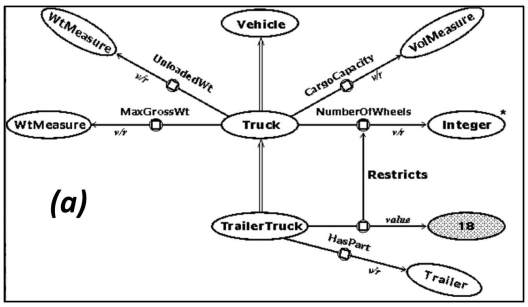
intelligence (AI) formalisms. They are known to be decidable fragments of first-order logic and they exist in multiple variants. *Feature logics* were developed as the constraint logic part of so-called unification grammars such as the head-driven phrase structure grammar (HPSG) (see http://hpsg.stanford.edu/ for more details). They differ from Description Logics (DL) [29] by the use of single-valued attributes whereas DL relies on multi-values ones. *Modal* and *description logics* are the leading logic-based formalisms. Also, some results from variations of modal logics (propositional dynamic logics, μ-calculus) have been translated into description logics [62, 234].

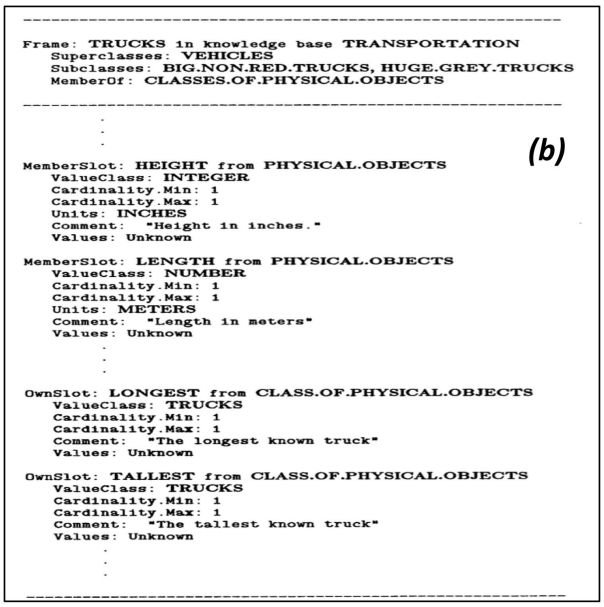
2.2.2 Description Logics Semantics

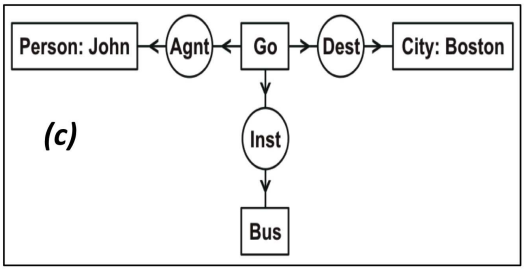
Description Logic Basics. Description logics are a family of logic based knowledge representation formalisms that describe domain in terms of concepts, roles and individuals [125]. Universal ($\forall$), existential ($\exists$), intersection ($\sqcap$), Union ($\sqcup$) and negation ($\neg$) operators are used to specify restrictions needed to make the language decidable with low complexity. In DL, semantics are defined by *interpretations*. An interpretation $\mathcal{I}$ is defined as followed:

$\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ where ($\Delta^{\mathcal{I}}$ is the domain of interest (non-empty set) and, $\cdot^{\mathcal{I}}$ is an interpretation function that maps:

- Concept name C : a subset $C^{\mathcal{I}}$ of $\Delta^{\mathcal{I}}$,
- Role name R : a binary relation $R^{\mathcal{I}}$ over $\Delta^{\mathcal{I}}$ and,
- Individual name x : an instance $x^{\mathcal{I}}$ of $C^{\mathcal{I}}$







E.g.,

$M \models \exists x.\mathrm{Cat}(x)$

$M \not\models \forall x.\mathrm{Cat}(x)$

(d)

$M \not\models \exists x.\mathrm{Cat}(x) \wedge \mathrm{Mat}(x)$

$M \models \forall x.\mathrm{Cat}(x) \rightarrow \mathrm{Animal}(x)$

$M \models \forall x.\mathrm{Cat}(x) \rightarrow (\exists y.\mathrm{Mat}(y) \wedge \text{sits-on}(x, y))$

$D = \{a, b, c, d, e, f\}$

$\mathrm{Felix}^I = a$

$\mathrm{MyMat}^I = b$

$\mathrm{Cat}^I = \{a, c\}$

$\mathrm{Mat}^I = \{b, e\}$

$\mathrm{Animal}^I = \{a, c, d\}$

$\text{sits-on}^I = \{\langle a, b\rangle, \langle c, e\rangle\}$

Figure 2.2: Illustrations of leading knowledge representation formalisms. (a) Conceptual definition of the Truck domain using a *Semantic network*, (b) Truck domain as a *Frame*, (c) Conceptual *(description) graph* of the statement "John is going to Boston by bus" [255], (d) First order logic (FOL)*description* of the cat domain and its interpretation [127]

Concepts, roles and individuals are respectively equivalent to FOL unary predicates, binary predicates and constants. Considering two concepts C_1 and C_2, a relation R, the interpretation function above extends to concept expressions as summarized in Table 2.1.

A more detailed definition of DL constructors can be found in Appendix A. Concepts, roles and individuals build up to the DL knowledge base $\mathcal{K}\langle\mathcal{T}, \mathcal{A}\rangle$ of a domain D. Here, $\mathcal{T}$ is a set of terminological axioms or *Tbox*, $\mathcal{A}$ is a set of assertional ax-

Name	Expressions(DL)		Interpretation (FOL)
Intersection	$(C_1 \sqcap C_2)^{\mathcal{I}}$	=	$C_1^{\mathcal{I}} \cap C_2^{\mathcal{I}}$
Union	$(C_1 \sqcup C_2)^{\mathcal{I}}$	=	$C_1^{\mathcal{I}} \cup C_2^{\mathcal{I}}$
Negation(concept)	$(\neg C_1)^{\mathcal{I}}$	=	$\Delta^{\mathcal{I}} \backslash C_1^{\mathcal{I}}$
Nominal	$\{x\}^{\mathcal{I}}$	=	$\{x^{\mathcal{I}}\}$
Existential quantifier	$(\exists R.C_1)^{\mathcal{I}}$	=	$\{x \mid \exists y.\langle x, y\rangle \in C_1^{\mathcal{I}}\}$
Value restriction	$(\forall R.C_1)^{\mathcal{I}}$	=	$\{x \mid \forall y.(x, y) \in R^{\mathcal{I}} \Rightarrow y \in C_1^{\mathcal{I}}\}$
Unqualified Number restriction ($\leq$)	$(\leq nR)^{\mathcal{I}}$	=	$\{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \leq n\}$
Unqualified Number restriction ($\geq$)	$(\geq nR)^{\mathcal{I}}$	=	$\{x \mid \#\{y \mid \langle x, y\rangle \in R^{\mathcal{I}}\} \geq n\}$
Inverse(relation)	$(\forall R^{-})^{\mathcal{I}}$	=	$\{(x, y) \mid (y, x) \in R^{\mathcal{I}}\}$

Table 2.1: Sample basic description logic (DL) constructors

ioms or *Abox*, x and y are individuals belonging to D. With respect to the modeling languages features, DL distinguish themselves by the non-finiteness of the domain and the open-world assumption of the knowledge modeling.

The DL Family There exists numerous DLs formalisms with various level of expressiveness. They are differentiated mostly base on what concept and role operators and, concept and role axioms are allowed and how they are used in the language. The $\mathcal{ALC}$ is the smallest propositionally closed DL. Expressions in this DL can be fully defined using five concept operators ($\sqcap, \sqcup, \neg, \exists, \forall$), two concept axioms ($\sqsubseteq, \equiv$). No role axioms are authorized and only atomic roles are allowed. For instance, a "great researcher" (vague in plain English) is a concept that can be formally described as follows.

`GreatResearcher` $\equiv$ `Person` $\sqcap$ $\forall$`hasAward.(ResearchAward` $\sqcup$ $\exists$`isPrestigiousAward.NobelPrize)`

In this example, the concept has been defined using two atomic concepts (`Person` and `ResearchAward`) two atomic roles (`hasAward`, `isPrestigiousAward`) and two operators (intersection and union) interpreted as set operations. Given that the notation $\forall R.C$ characterizes the set of individuals that are in the relationship R with

individuals belonging to the set denoted by the concept C, the expression following the "·" is a "role filler" to the atomic role `hasAward`.

In the $\mathcal{EL}$ DL also known as the sub-Boolean DL, there is no universal quantifier ($\forall$). Thus, the above statement can't be written as is. The best one can do is to define a "researcher" as a person who conducts research as follows.

$$\texttt{Researcher} \equiv \texttt{Person} \sqcap \exists \texttt{conductsResearch}$$

This shows the expressiveness limits of some DLs and highlights the need for strategic and selective extensions to resolve this problem. This is particularly critical in ensuring the decidability of the knowledge modeling language supported by the DL under consideration. Appendix B illustrates some of these extensions to the $\mathcal{ALC}$ or $\mathcal{S}$. Some of them are as follows.

- Role hierarchy $\mathcal{H}$: e.g., $\texttt{hasAcademicAward} \sqsubseteq \texttt{hasAward}$
- Role box (composition)$\mathcal{R}$: e.g., $\texttt{hasPublication} \circ \texttt{hasWonPrize} \sqsubseteq \texttt{hasAward}$
- Nominal/singleton $\mathcal{O}$: e.g., $\{\texttt{Petnga}\}$
- Inverse roles $\mathcal{I}$: e.g., $\texttt{isPrestigiousAwardedTo} \equiv \texttt{hasAward}^{-}$
- Number restrictions $\mathcal{N}$: e.g., $\geq 3\texttt{hasAward}$ (more than awards)
- Qualified number restrictions $\mathcal{Q}$: e.g., $\leq 2\texttt{hasAward.NobelPrize}$ (awarded less than 2 Nobel prizes)

As an illustration, the $\mathcal{SHIQ}$ DL can be determined as follows. $\mathcal{SHIQ} = \mathcal{S}$ + role hierarchy($\mathcal{H}$) + inverse roles($\mathcal{I}$) + $\mathcal{QNR}$

Inferencing Services. The knowledge base $\mathcal{K}$ introduced above can be extended with the addition of new facts. However, one should first make sure that $\mathcal{K}$ is well-

constructed and is inconsistency free. Thus, it should be sound with respect to known basic inferencing tasks as illustrated in Figure 2.3 and formally defined in Appendix C.

Subsumption. This reasoning service checks if the knowledge in the Tbox $\mathcal{T}$ of $\mathcal{K}$ is correct. It also establishes hierarchy among concepts i.e., $C_1 \sqsubseteq_{\mathcal{K}} C_2$?

Equivalence. It checks for redundancy between elements in the knowledge base $\mathcal{K}$ and establishes equivalence between representations in the Tbox. In other words, the question answered here is $C_1 \equiv_{\mathcal{K}} C_2$?

Consistency. The main goal of this reasoning task is to ensure that the knowledge in $\mathcal{K}$ is meaningful. In other words, $C_1 \equiv \bot$.

Instantiation. This task consists of checking if an individual i is an instance of a class C_1. This can be written as follows: $i \in_{\mathcal{K}} C_1$.

Satisfiability. It consists of checking the consistency of either a concept, an Abox or the knowledge base $\mathcal{K}$. As defined in Appendix C, the satisfiability of concepts and the Abox$\mathcal{A}$ are checked with respect to the Tbox $\mathcal{T}$. The knowledge base $\mathcal{K}$ is satisfiable $\iff \exists M$ s.t. $M \models \mathcal{K}$, where M is a model. Tableaux algorithms used to test satisfiability of the knowledge base. An interesting resource to figure out the complexity of reasoning in description logics can be found at *http://www.cs.man.ac.uk*.

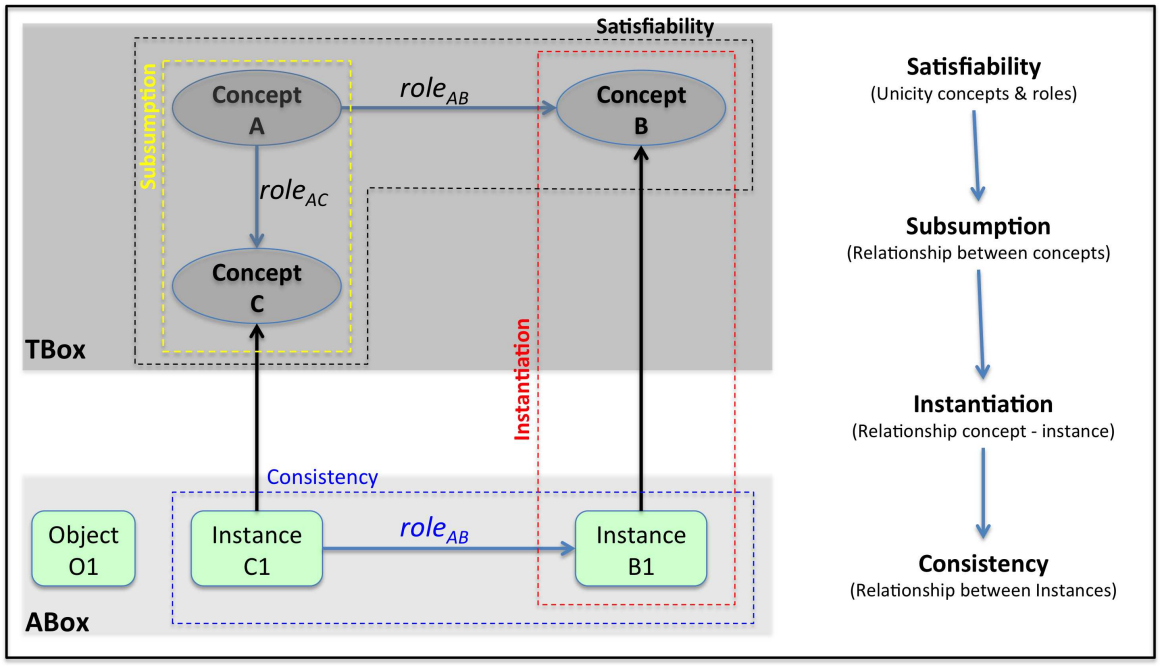


Figure 2.3: Illustrations of foundational DL reasoning algorithms.

2.2.3 Ontologies and Ontological Languages

Ontologies. The quest of identifying entities and types of entities that exist has led earlier Philosophers to ontology, which they defined as the study of "being" or "existence" and their basic categories. They have classified ontologies in four types base on their degree of abstraction and field of application [215]. *Upper ontologies* formally define high level meta concepts while *domain ontologies* focus on concepts relevant to a given, specific subject area or domain. *Interface ontologies* formally define the juncture of two disciplines or domains and *process ontology* describe process domains (business or engineering).

Over the years, these research efforts have provided means for humans to develop and access semantic contents beyond simple definitions and taxonomies.

However, this has been harder to achieve in a very efficient way for computers and engineering systems [126]. In the scientific community, an ontology is an engineering artifact that provides explicit specification of the intended meaning of a vocabulary used to describe a given domain. Constraints in the specifications help capture background knowledge (which does not have to be complete) about the domain. For the ontology to be fully useful for modeling and reasoning applications, it has to (1) capture the shared understanding of the domain of interest and, (2) provide a formal and machine readable model of that domain knowledge. Ontologies have been shown effective in capturing and formally represent reusable knowledge in various domains such as biology [238], healthcare and medicine [97], geography [99], agriculture [251] and defense [148].

Ontology Languages. They are formal, declarative languages used to build ontologies. Knowledge representation formalisms provide the appropriate semantics needed by ontology languages to effectively and precisely capture and represent knowledge with regard to the chosen theory in a human-readable way. Expressiveness and inferencing are central in determining the capabilities of ontological languages. However, a trade-off between these two elements is needed as one comes at the cost of the other. In the ontology community, a distinction is generally made between earlier languages qualified as traditional such as Ontolingua or F-logic from the ones that use the Extensible Markup Language (XML) scheme to encode knowledge such as the RDF. From a structural stand point, frame-based languages (e.g.: F-logics, Knowledge Machine programming language,..) are dis-

tinguished from description-logic based languages (e.g.: KL-ONE, Racer,..) and first-order logic -based languages (e.g.: common logic, CycL,...). Languages in the first group are partially or completely based on frame knowledge representation formalisms as introduced in Section 2.2.1. DL-based languages extend frame-based languages while keeping the focus on making the languages both human and machine readable. The last group of languages enable both the formulation of expressions in FOL as well as arbitrary predicates. A comprehensive comparative study of ontological languages and their capabilities can be found at [58].

2.3 Semantic Extensions and Support for Web-Based Reasoning

This section introduces formalisms for the capture and representation of knowledge, suitable for decision making support in CPS development. Traditional approaches to knowledge representation and reasoning stem from artificial intelligence and classical logics, which may or may not be decidable. Our focus is on methods that are computationally decidable. As such, we make extensive use of description logics and its various extensions.

2.3.1 Description Logics Extensions for the Web Ontology Language

In order to build models that address the challenges identified in Chapter 1, CPS applications need to be backed by ontologies that have well-defined semantics and support for formal reasoning. DLs provide these formal foundations to the web ontology language [201]. In fact, the semantics of the OWL language can be

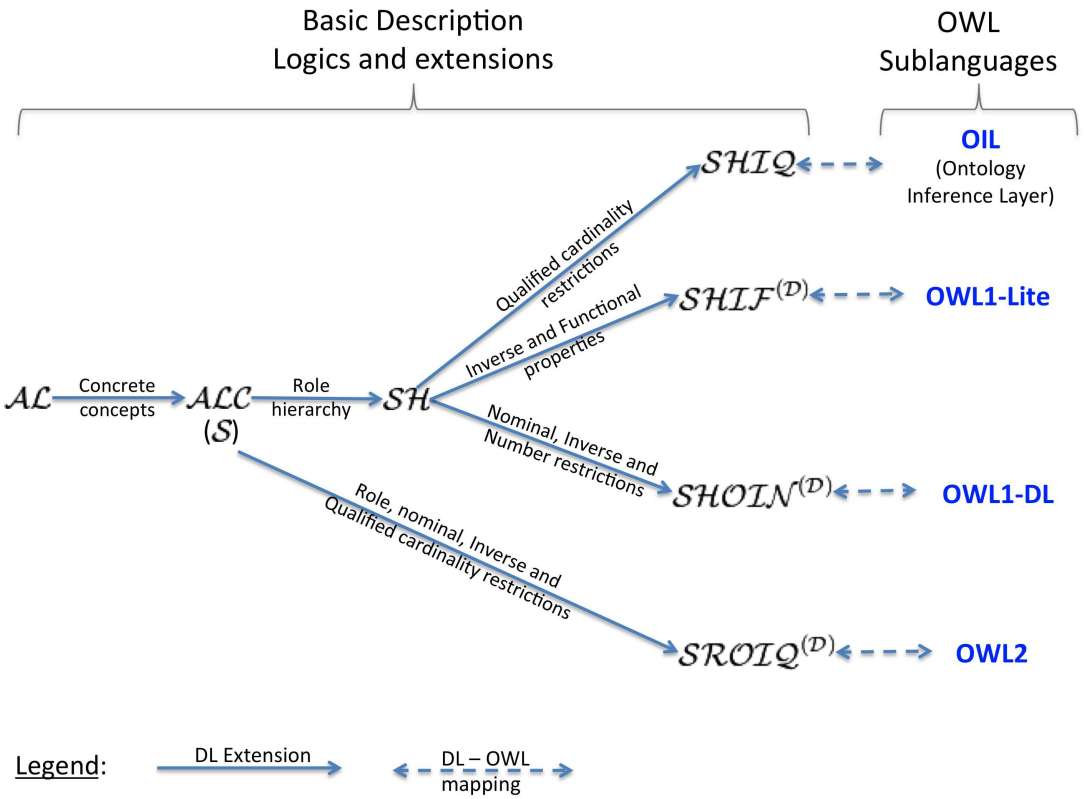


Figure 2.4: Description Logics formalism extensions for the Web Ontology Language

defined through a translation into an expressive DL. However, as pointed out by Baader and co-workers [28], the $\mathcal{ALC}$ extensions (see Appendix A) are incapable of efficiently supporting OWL because important pieces are missing. Bridging this gap requires a certain number of extensions including support for role Hierarchy ($\mathcal{H}$), Nominals ($\mathcal{O}$), Inverse and transitive roles($\mathcal{I}$), Cardinality/Number restriction ($\mathcal{N}$), Qualified number restrictions($\mathcal{Q}$), Role restrictions($\mathcal{R}$) and Concrete domains. These extensions are briefly defined along with illustrative examples in Appendix B.

Figure 2.4 shows how these extensions to $\mathcal{ALC}$ DL can be organized and

mapped to semantics for the OWL sub-languages. To that extend, OWL 2 (standardized in 2009) overcomes a number of weaknesses (e.g., relational expressivity, syntax deficiencies, species definitions) in OWL 1 [102]. Tapping into this potential for efficient modeling and decision support for CPS-based applications requires effective and decidable reasoning capabilities as enablers. We briefly introduce in the next section the reasoning infrastructure needed to that aim.

2.3.2 Reasoning Support for $\mathcal{SROIQ}$ - based Ontologies

When the relevant set of axioms are applied to a specific DL-based ontology, the result is a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ for the domain being modeled. However, this is half of what we later need. This foundation needs to be completed with a reasoner that can derive, through inferencing, additional facts about the concepts of the domain of interest. Among the key reasoning services needed, are satisfiability, subsumption, equivalence and disjointness. These services are formally defined in Appendix C. Also, with regard to the $\mathcal{SROIQ}$-DL which is mapped to OWL 2 (see Figure 2.4) there is a need to formally establish the decidability of this DL. Thus, proposition 1.1 builds on the definitions introduced to establish the satisfiability of the TBox while Lemma 2 ensures the elimination of the ABox for the purpose of simplifying the complexity of the reasoning process. Horrocks and co-authors [128] use these preliminary results to construct and describe an algorithm that decides the satisfiability and subsumption of $\mathcal{SROIQ}$ as stated by Theorem 4. Hence, given the mapping in Figure 2.4, this theorem ensures the decidability of OWL 2 DL, the

language of development of the ontological framework that we introduce in Chapter 5.

2.4 Working with Semantic Web Technologies

2.4.1 Low-Level Technologies (IRI and UNICODE)

At the bottom of the semantic web stack, unicode provides 16-bit support for multiple languages, and internationalized resource identifiers (IRI) provide a means for the unique identification of resources on the Web. Unicode enables the multi-language representation and handling of texts.

2.4.2 Extensible Markup Language (XML)

The extensible mark-up language provides a syntactic basis for data exchange and is both human and machine interpretable. XML technology has two aspects. First, it is an open standard which describes how to declare and use simple tree-based data structures within a plain text file (human readable format). XML is a meta-language (or set of rules) for defining domain- or industry-specific markup languages. Within the systems engineering community, for example, XML is being used in the implementation of AP233, a standard for exchange of systems engineering data among tools [186]. A second key benefit in representing data in XML is that we can filter, sort and re-purpose the data for different devices using the Extensible Stylesheet Language Transformation (XSLT) [268, 292].

2.4.3 Resource Description Framework (RDF)

While XML provides support for the portable encoding of data, it is limited to information that can be organized within hierarchical relationships. This can be a problematic situation for XML as a synthesized object may or may not fit into a hierarchical (tree) model. A graph, however, can, and thus we introduce the Resource Description Framework (RDF).

RDF is a graph-based assertional data model for describing the relationships between objects and classes (i.e., data and metadata) in a general but simple way, and for designating at least one understanding of a schema that is sharable and understandable. The graph-based nature of RDF means that it can resolve circular references, an inherent problem of the hierarchical structure of XML. An assertion is the smallest expression of useful information. RDF captures assertions made in simple sentences by connecting a subject to an object and a verb, as shown in Figure 2.5.

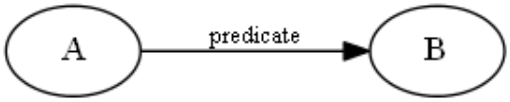


Figure 2.5: Example of RDF triple where node A is a subject, "predicate" is a verb, and node B is an object.

In practical terms, English statements are transformed into RDF triples consisting of a subject (this is the entity the statement is about), a predicate (this is the named attribute, or property, of the subject) and an object (the value of the named

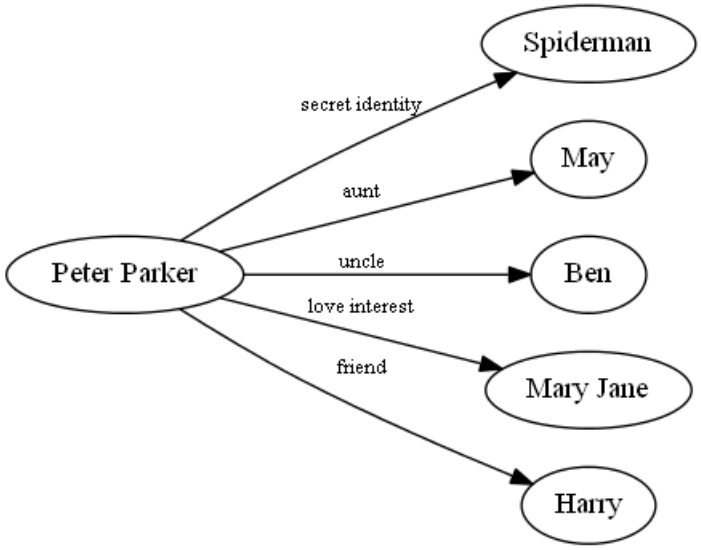


Figure 2.6: An RDF graph of relationships important to Spiderman.

attribute). Subjects are denoted by a IRI. Each property will have a specific meaning and may define its permitted values, the types of resources it can describe, and its relationship with other properties. Objects are denoted by a "string" or IRI. The latter can be web resources such as requirements documents, other Web pages or, more generally, any resource that can be referenced using a IRI (e.g., an application program or service program).

A set of related statements constitute an RDF graph. RDF graphs can be used to model a wide variety of relationships, including those among friends, location data, business data, and show information about a restaurant and a movie [237]. Figure 2.6 illustrates, for example, a graph model of relationships relevant to Spiderman.

Limitations of RDF. Unfortunately, RDF is unable to capture vital knowledge attributes such as existence and cardinality or localized range and domain con-

straints as well as richer properties such as transitivity, inverse or symmetrical properties [126]. This makes it weaker to describe resources in sufficient detail and difficult in use to support reasoning as introduced in Section 2.2.2. The Ontology Inference Layer (OIL) and DARPA Agent Markup Language (DAML) were developed separately and respectively by European and American researchers to address the weaknesses of RDF. They were subsequently merged into DAML+OIL which became the foundation of OWL [129].

2.4.4 The Web Ontology Language (OWL)

Pathway from RDF to OWL. The Web Ontology Language (OWL) is a DL-based knowledge representation language for constructing ontologies. OWL adds expressiveness to the Semantic Web for knowledge representation, information content processing and machine interoperability.

One key driver of the development of OWL was for the language to support the creation of extensible, ease of use, ease of querying ontologies that are compatible with the world wide web (WWW) as recommended by the World Wide Web Consortium (W3C) [1]. Earlier effort in that direction led to the development of the RDF which, along with its schema extension (RDFS), has later evolved into OWL as shown in Figure 2.7.

Structure and Family of OWL. OWL is based on the basic features of RDF introduced above but it strengthens it by adding structure and vocabulary for describing properties and classes. They enable richer property definitions(e.g.: transitivity),

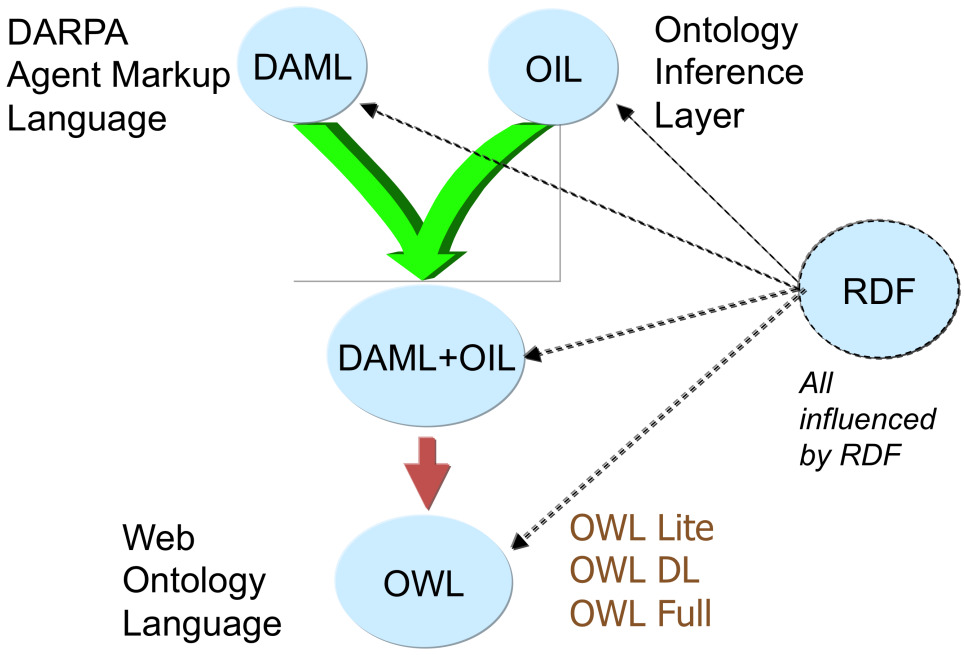


Figure 2.7: The making of the web ontology language (OWL): From the resource description framework(RDF) to OWL (adapted from [129])

class property restrictions(e.g.: someValuesFrom), equality between classes(e.g.: sameAs) and relations between classes (complementOf). The additional capabilities allow ontological systems to use reasoning structures and infrastructure to infer new facts (triples) from existing ones with FOL as baseline mathematical, formal foundation. As a matter of fact, Horrocks [126] points that OWL is a Web-friendly syntax for $\mathcal{SHOIN}$ DL. As an illustration, the DL concept of "great researcher" introduced in Section 2.2.2 can be translated in OWL as follows (see Figure 2.8).

The family of OWL encompasses three languages distinguished by their increasing expressiveness. *OWL Lite* allows the expression of simple syntax and constraints but inferencing is more tractable using this version. *OWL DL* has a human-friendly syntax, inferencing is decidable and the language is computationally complete. *OWL Full* ensures full compatibility with RDF and RDFS languages

```
<owl:Class>
    <owl:intersectionOf rdf:parseType=" collection">
      <owl:Class rdf:about="#Person"/>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasAward"/>
        <owl:allValuesFrom>
          <owl:unionOf rdf:parseType=" collection">
            <owl:Class rdf:about="#ResearchAward"/>
            <owl:Restriction>
              <owl:onProperty rdf:resource="#isPrestigiousAward"/>
              <owl:someValuesFrom rdf:resource="#NobelPrize"/>
            </owl:Restriction>
          </owl:unionOf>
        </owl:allValuesFrom>
      </owl:Restriction>
    </owl:intersectionOf>
  </owl:Class>
```

Figure 2.8: Example of a formal definition of a "great researcher" in OWL.

however, the cost is that there is no guarantee in the validity of all computed statements [1]. Also, the most recent version of the language i.e., OWL2 comprises three independent profiles (or sub-languages) that restrict its structure in different ways : expressiveness (OWL2 - EL), Querying (OWL2 - QL) and Reasoning (OWL2 - RL). We note that previous versions: i.e., OWL1 Lite (limited expressiveness), OWL1 DL (decidable with support for DL) and OWL1 Full (full expressiveness) can also be viewed as profile of OWL2 [290].

2.5 Working with Jena and Jena Rules

Not all technologies on the semantic web are standardized. Some are emergent ones that are used mostly for horizontal and vertical integration of multiple layers of the stack. Generally speaking, they are Application Programming Inter-

faces (API) used to complete integration tasks.

2.5.1 Jena

Jena [4] is an open source Java framework for building semantic web and linked data applications. It provides APIs for RDF, triple store and OWL (ontology and inference). Jena uses a rule-based reasoning approach, which is the classic technique to logic-based reasoning where the knowledge-based system is developed by deduction, induction, abduction or choices from a starting set of data and rules. A unifying logic, such as the DL, is needed for horizontal integration of top layers of stacks and provide the rigorous, formal support needed by system (CPS) models. The latter are the result of the top level, i.e., the application layer which allows users to visualize and interact with whatever underlying semantic platform supporting the application.

2.5.2 Jena Rules

The Jena inference subsystem is designed to allow a range of inference engines or reasoners to be plugged into Jena. Jena Rules is one such engine. Reasoners provide a means to derive additional RDF assertions which are entailed from some base RDF together with any optional ontology information and the axioms and rules associated with the reasoner. Jena Rules use facts and assertions described in OWL to infer additional facts from instance data and class descriptions. Such inferences result in structural transformations to the semantic graph model.

Remark 2.1. *(Ontological Tools).* The development of ontology applications is facilitated with research and commercial tools such as Oiled for DAM+OIL ontologies (*http://oiled.semanticweb.org/building/*) and Protege for OWL ontologies [124]. Others, such as WebODE are able to translate, import, and export ontologies in multiple languages (e.g., RDF,RDFS, OIL, DAML+OIL,etc.) [59]. Concurrent work in the development of reasoners includes: Pellet [243], RacerPro [8], FaCT++ [9] and Hermit [10].

2.6 Case Study: Semantic Modeling of Family Dynamics

This case study examines the work of Austin, Delgoshaei and Nguyen [24] from the perspective of basic ontology- and rule-based modeling of systems with Jena and Jena rules. A simplified semantic model of a family is defined by ontologies (Jena) and constrained by rules (Jena Rules). Once the family model has been assembled, the graph of family individuals and relationships will evolve in response to events.

2.6.1 Family Ontology and Graph (Jena)

Figure 2.9 illustrates the appeal of behavior modeling with ontologies and rules [139, 167, 240]. The upper right-hand side of the figure shows the relationship among classes and properties in a simplified family ontology. A person has properties: `hasAge`, `hasWeight`, and `hasBirthDate`. `Male` and `Female` are subclasses (specializations) of class `Person`. `Boy` is a specialization of `Male`. A `Child` is a

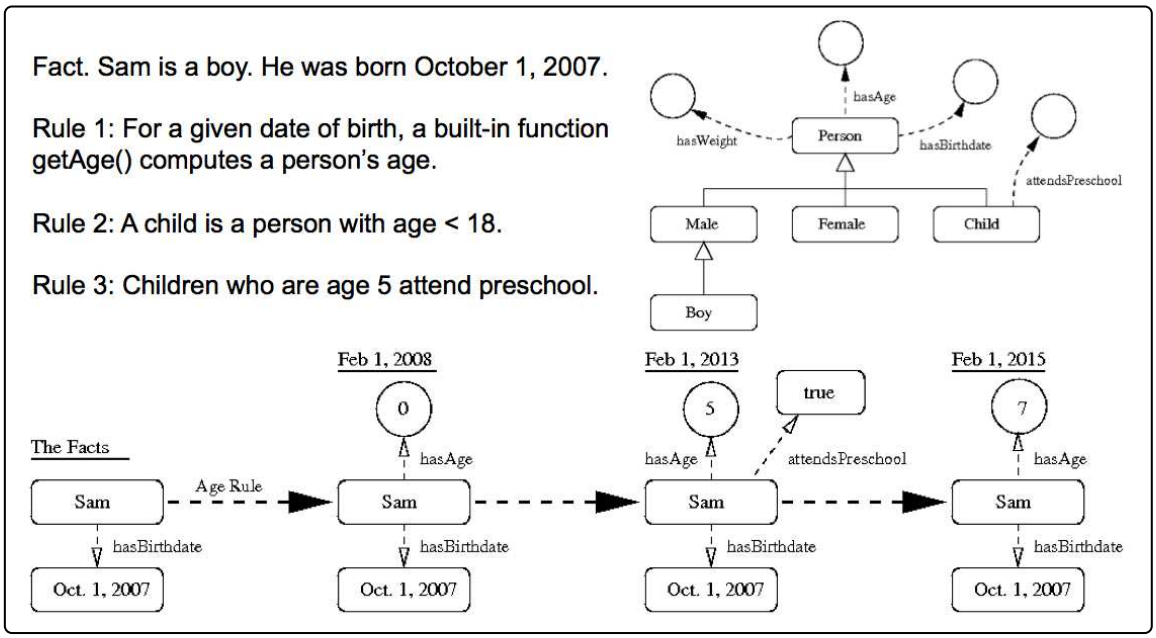


Figure 2.9: Simplified framework for modeling with ontologies and rules.

`Person` who may (or may not) attend Preschool.

The abbreviated fragment of code:

```
// Define classes ...

person = model.createClass( ns + "Person");
male   = model.createClass( ns + "MalePerson");
boy    = model.createClass( ns + "Boy");

// Define relationships among classes ...

person.addSubClass (   male );
male.addSubClass (  boy );

// Create data properties for the class Person ...

hasAge = model.createDatatypeProperty( ns + "hasAge");
hasAge.setDomain(person);
hasAge.setRange( XSD.integer );

hasBirthDate = model.createDatatypeProperty( ns + "hasBirthDate");
hasBirthDate.setDomain(person);
hasBirthDate.setRange( XSD.date );
```

demonstrates the definition of the family ontology classes, their assembly into a hierarchy, and definition of data properties for the class `Person`. The data property `hasAge` is an integer. The data property `hasBirthDate` is a date. Notice that since `Boy` is a subclass of `MalePerson`, and `MalePerson` is a subclass of `Person`, boys automatically have the properties age and birthdate through class hierarchy inheritance.

The next step is to define family individuals, the data associated with each individuals, and the relationship of one individual to other individuals in the family. The fragment of code:

```
// Namespace for the family ontology ...

String ns = "http://austin.org/family#";

// Create ontology model (a graph) ...

OntModel model = ModelFactory.createOntologyModel();

// Add "Sam" to the family graph model ...

Individual sam = boy.createIndividual( ns + "Sam" );
model.add ( sam );

// Create statement: Sam's birthdate is 2007-10-01.

Literal bdate = model.createTypedLiteral( "2007-10-01", XSDDatatype.XSDdate );
Statement cbd = model.createStatement( sam, hasBirthDate, bdate );
model.add ( cbd );
```

establishes a name space for the family ontology, creates a graph model for the storage of individuals and their data and object properties, and then creates an Individual model for `Sam` and a data property statement for his date of birth. Jena provides very powerful facilities for querying the graph model, subject to a wide

range of search criteria.

2.6.2 Event-Driven Graph Transformations (Jena Rules)

The upper left-hand side of Figure 2.9 shows one fact and three rules. `Sam` is a boy born October 1, 2007. Given a birthdate and a current time, a built-in function `getAge()` computes Sam's age. Further rules can be defined for when a person is also a child and when children attend Preschool. The schematic along the bottom of Figure 2.9 shows the evolution of a graph defining the properties of `Sam` as a function of time. The abbreviated fragment of code:

```
@prefix af:  <http://austin.org/family#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

// Rule 01: Propagate class hierarchy relationships ....

[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y) ->
          [ (?a rdf:type ?y) <- (?a rdf:type ?x)] ]

// Rule 02: Compute and store the age of a person ....

[ GetAge:    (?x rdf:type af:Person) (?x af:hasBirthDate ?y)
             getAge(?y,?z) -> (?x af:hasAge ?z) ]
[ UpdateAge: (?a rdf:type af:Person) (?a af:hasBirthDate ?b)
             (?a af:hasAge ?c) getAge(?b,?d) notEqual(?c, ?d) ->
             remove(2) (?a af:hasAge ?d) ]
```

is taken from the Jena Rules for the family ontology. The first rule propagates class hierarchy relationships. The second set of rules serves two purposes. First, given an individuals data of birth, the *GetAge* rule computes their age and inserts it into the semantic model via the `hasAge` data property. When a person has a birthday, the *UpdateAge* rule removes the old age from the graph and inserts the new age.

Chapter 3: Semantic Modeling of Time

3.1 Introduction

Timed systems are those where timing and scheduling of events (i.e., logical results of computation) are relevant to correct operations [123]. In hard real-time systems, correct operation depends of satisfaction of hard time-ordered deadlines. In soft real-time systems, correct operation depends on satisfaction of "average time" constraints.

Formal approaches to CPS design need precise, ambiguity-free, and consistent representations of time in order to support the proper ontological modeling of this domain. However, after 2,500 years of research on the nature of time there is not much to show for, beside clocks that can measure time and help locate events, their order of occurrence and durations. There remain many critical, semantically-related, unresolved issues including the fundamental questions of "What time actually *is*" and "Which aspects of time are subjective or mind-dependent" [68].

This chapter examines methods for the semantic modeling of time. The scope of investigation covers a review of methods capable of representing and working with time computations that are both quantitative and qualitative, continuous and

discrete. Models and properties of time are visited in Section 3.2 and the ontological descriptions of this domain are conducted in Section 3.3. Methods and tools for qualitative temporal reasoning have been developed over the past three decades and we revisited them in Section 3.4. They allow applications to manage coarse-grained causality, action and change. A notable and highly influential example is Allen's interval calculus that is illustrated in Section 3.5.

3.2 Models and Properties of Time

For formal approaches to the verification of CPS to work, we need formal models to capture the appropriate granularity of time.

3.2.1 Discrete versus Dense Time

A first natural categorization of formalisms dealing with time-dependent behavior is whether such a model is a discrete or dense set. From a mathematical standpoint, notions of discrete and dense time (see Figure 3.1) can be expressed as follows [87]:

- A discrete set consists of isolated points (e.g., the integers 0, 1, 2, ...) separated by regular intervals.
- A dense set (ordered by $<$) is such that for every two points t_1, t_2, such that $t_1 < t_2$, there is always another point in-between (i.e., $t_1 < t_3 < t_2$).

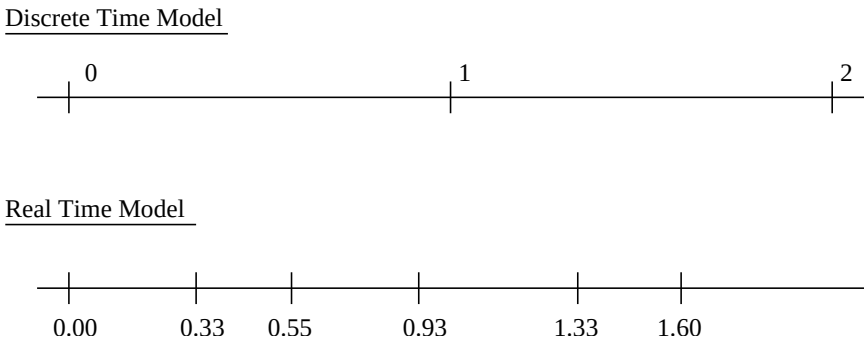


Figure 3.1: Schematic of discrete and dense time models.

The main advantage of discrete-time representations is conceptual simplicity. In a discrete time domain time is advanced by discrete steps. Specific tick events are used to model the advance of one time units, and Events can only happen at integer time values. The delay between any two events is always a multiple of the minimal delay of one time unit [239].

Modeling time as a dense (continuous) process means that time is modeled by real numbers and changes in state can happen at any point in time. The delay between two events can be arbitrarily small, Dense models of time provide a more adequate representation of reality, particularly asynchronous systems. However, because dense time implies an infinite number of possible states, timed systems are modeled symbolically rather than explicitly.

In practical terms, the challenge in working with time is complicated by the tendency of analytical procedures to mix-and-match dense/discrete formalisms, and to sometimes rely on qualitative relationships among entities involving time. For example, differential equations are normally stated with respect to real variable domains, whereas difference equations are defined on integers. Also, in control sys-

tems, *event-triggered* and *time-triggered* are two control paradigms that can make use of both discrete and continuous models of time, depending on the criticality of the application [140]. In event-triggered real-time systems, activities are initiated as a result of significant and identifiable events. However, in time-triggered systems, activities are started at predefined points in real-time. Computing devices are formalized through discrete models when their behavior is paced by a clock, so that it is natural to measure time by counting clock, ticks, or when they deal with (i.e., measure, compute, or display) values in discrete domains. Qualitative evaluation of relationships between events in time are expressed in terms of "before or after" but omitting details such as "how much before or how much after."

3.2.2 Time Instants and Intervals

Instants in the dense model of time are isomorphic to the rational numbers: between any two instants there is always another. Continuous models of time are isomorphic to the real numbers, i.e., they are dense and also, unlike the rational numbers, without gaps.

3.2.3 Qualitative Descriptions of Time

The frequent under-specification of time in natural language expressions constitutes the leading source of uncertainty in temporal knowledge representation. Vagueness in the granularity of time in temporal statements stems from the multiple forms temporal references can take. As an illustration, let's consider temporal

references in various statements related to the Rio Olympics.

The Rio Olympics started ...

(a) ... at 7:00 PM (ET) on Friday, August 5, 2016.

(b) ... in the evening of Friday, August 5, 2016..

The Rio Olympics lasted ...

(c) ... 17 days.

(d) ... less than a month.

The Rio Olympics happened ...

(e) ... after Copa America Centenario.

(f) ... during summer.

Temporal entity references can be absolute (a) or relative (b). Similarly, temporal durations can either be absolute (c) or absolute (d). Moreover, the order in which events occur in the temporal domain can be specified in an order that can be certain (e) or uncertain (f). On the other hand, under the assumption that the minute is the granularity of our time measurement, statement (a) is fully specified otherwise it could not be the case; for instance if, instead of minute, the granularity of time is second. Removing the reference to the time zone (ET) will lead to the same conclusion.

3.2.4 Precedence Relations

Operators are needed to describe relationships among time intervals (e.g., looking forward, looking backward, contained within, separated ...etc). These relations can be described through precedence relations.

Relation	**Mathematical Representation**
Transitivity	For all x,y,z, if $x < y$ and $y < z$ then $x < z$.
Nonreflexivity	For all x, Not $(x) < x$.
Linearity	For all x,y, $x < y$ or $x = y$ or $x > y$.
Left Linearity	For all x,y,z, $y < x$ and $z < x$ implies $y < z$ or $y = z$ or $z < y$.
Begin	There exists an x and not a value of y such that $y < x$.
End	There exists an x and not a value of y such that $x < y$.
Predecessor	For all x, there exists y such that $y < x$.
Successor	For all x, there exists y such that $x < y$.

Table 3.1: Properties defining the temporal domain structure.

The properties "begin" and "end" state that the temporal domain is bounded in the past (future). The properties "predecessor" and "successor" show that the temporal domain is unlimited in the past (future). These relationships dictate the set of formulae that temporal logics can express.

3.3 Ontological Descriptions of Time

3.3.1 Temporal Theories and Calculus

Theories of Time. The diversity of views of meaning of time has led to various temporal theories in multiple domains from philosophy to physics passing by knowledge representations. Thus, we identify and analyze those theories along with corresponding temporal calculus in the context of CPS design. Time as point, interval, duration and dimension appear among the leading theories [108, 115].

i. **Time-point.** The notion of a point in time supports this temporal theory; this concept is sometimes assimilated to the one of a "position in a temporal coordinate system" which has no duration and is useful in locating events on the time-plenum.

ii. **Time-interval.** Pieces of time located on the temporal continuum (or time-plenum) serve as a basis for the temporal theory. Some researchers have also attempted to theorize *Temporal regions*. We'll put those efforts in this category of temporal theories with the rationale that an interval is a special type of region. Allen's temporal interval calculus [18] is an example of the interval perspective while time in the context of space-time region in the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) illustrates the region viewpoint [179].

iii. **Time-duration.** Constant amounts of time that can be compared and are distinctive from the length of the time interval are used to define time. Some researchers have extended duration calculus for conventional control theory [297].

iv. **Time-dimension.** Time is considered a physical dimension such as length, mass or voltage, with unit and physical properties as specified by Gruber and Olsen [108].

Selecting a Candidate Temporal Theory. Many engineering modeling paradigms have looked at those theories mostly for the purpose of physical systems modeling, merely considering time as simple quality of service parameter in the cyber world. This has to change as temporal concerns in complex, software-intensive systems such as CPS include guarantees that computations will be achieved within required response times [155, 156]. Therefore, appropriate theories and calculus should: (a) enable the appropriate granularity of time for system applications to be efficiently captured by formal models and, (b) be backed by sound temporal logics that can support practical temporal reasoning. Irrespective of the chosen theory and related calculus, one ought to be able to clearly express relationships between temporal entities. The ultimate goal is to later enable the reasoner to answer mereological (part-of), topological (connects) and logical ("rules-based") questions on the temporal domain. One would also like to capture other relevant aspects of time such as clock, calendar and temporal aggregates while adopting the appropriate granularity.

3.3.2 Specifications of Time

Researchers and practicians have long seek to develop easily understandable, reusable, flexible and formally defined ontologies of time. Whether they are developed as part of foundational, Upper-Level or stand alone ontologies, their usefulness in large scale knowledge system [77] and the Semantic Web applications has been shown very important [121]. Existing ontologies of time employ a combination of foundational theoretical primitives introduced in Section 3.3.1, but they are otherwise strongly influenced by the targeted need for which they were developed. The ontologies are usually structured using a combination of hierarchized classes, properties, axioms and instances constructed on above-mentioned primitives. As illustration, Hatala et al. [114] rely on a discrete time-space point model to store user paths in their real-time audio museum application. Gruninger [109] introduces time-point based axioms to support formalizing the process specification language (PSL). The time in this theory is based on totally ordered time-points and it supports branches for possible futures. In OWL-Time [2], Instant and Interval are basic mereological individuals, serving as foundational temporal entities (see Figure 3.2). Beside the selection of the temporal primitive and theory, the knowledge modeler has to make a certain number of decisions regarding the granularity of time, representation and use of calendars as well as temporal intervals (open vs close).

Despite sustained research efforts and resulting high variety of offers, a recent review and analysis of the paper collection of the TIME Symposia series (*http://time.di.unimi.it/*)

```
:Instant
a       owl:Class ;
rdfs:subClassOf :TemporalEntity .
:Interval
a       owl:Class ;
rdfs:subClassOf :TemporalEntity .
:TemporalEntity
a       owl:Class ;
rdfs:subClassOf :TemporalThing ;
owl:equivalentClass
[ a       owl:Class ;
owl:unionOf (:Instant :Interval)] .
```

Figure 3.2: Definition of instant, interval and temporal entity in OWL-Time.

from 1994 at 2014 shows that existing ontologies of time still do not meet all the needs of the Artificial intelligence and Semantic web communities [72]. Those needs have been formalized in the form of an ontology for which features of time satisfy a "synthetic theory" defined by the TIME community. The profile of this baseline theory are defined based on the attributes of selected concepts areas in the taxonomy of temporal features shown in Figure 3.4.

The gaps identified are the lack of coverage by current ontologies of the following critical aspects:

(i) *Density of time* which is a critical tradeoff parameter between model expressive power and its computational complexity,

(ii) *Relaxed linearity* for expressing parallel independent time lines,

(iii) *Scale factors* which would enable time to run with different "velocities",

(iv) *Periodicity* of temporal structures such as subintervals,

(v) *Measures* formats (e.g.: date/time) and clocks.

3.3.3 Allen's Temporal Intervals Calculus

Allen's temporal interval calculus (ATIC) is an interval-based temporal theory developed and introduced by James F. Allen in the early eighties [18, 19]. He identifies and specifies thirteen (13) relationships between any ordered pair of "convex" time intervals as the core of his Interval Algebra. The main seven (7) relationships are illustrated in Figure 3.3; Six (6) inverse relations also exist. The strength of Allens interval algebra resides in its capability to manipulate interval and express temporal properties and their evolution over those intervals. At the core of this algebra is the relationship between time intervals. Thus, given two time intervals I_1 and I_2, a time-point t and a proposition ϕ, we might ask a variety of questions over the time domain such as: **(1) Mereological or part-of questions** (e.g., Is the interval I_1 a sub-interval of I_2? Does t occur within I_1? Is the interval I_1 equals to I_2? What interval represents the temporal intersection of I_1 and I_2? Does interval I_1 contains interval I_2?); **(2) Topological or connects questions** (e.g., Does interval I_1 happens before or after interval I_2? Do intervals I_1 and I_2 meet? Do intervals I_1 and I_2 start and/or end at the same instants?) and, **(3) Logical or rules-based questions** (e.g., Does the proposition ϕ hold within the interval I_1? If ϕ holds during the interval I_1 does it hold during I_2 too? Does the proposition ϕ hold before or after the interval I_1?).

Hobbs & Pan [121] provide formal definitions of these interval relations in

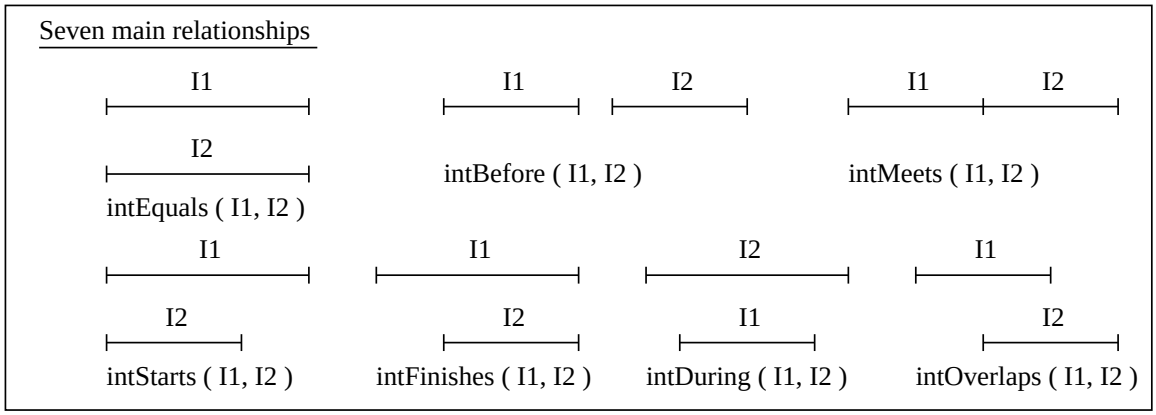


Figure 3.3: Allen's temporal intervals: Seven main relationships among intervals of time

terms of $before$ relations among their beginning and end points, which exist, given their proper nature. Moreover, these intervals are closed i.e. if we consider intervals over a time domain (T) with $\{beginAt, endAt, t\} \subseteq (T)$, then t is within the *closed* interval $[beginAt, endAt]$ if $beginAt \leq t$ and $t \leq endAt$. For instance, if I_1 and I_2 are two time intervals, the above *intOverlaps* relationship is defined as follows.

$$intOverlaps(I_1,I_2) \equiv [ProperInterval(I_1) \wedge ProperInterval(I_2)]$$

$$\wedge\ (\ \exists\ t_2,\ t_3)[ends(t_2,\ I_1\ ) \wedge begins(t_3,\ I_2\ ) \wedge before(t_3,\ t_2)$$

$$\wedge\ (\ \forall\ t_1\ )[begins(t_1,\ I_1\ ) \Rightarrow before(t_1,\ t_3\ )]$$

$$\wedge\ (\ \forall\ t_4\ )[ends(t_4,\ I_2\ ) \Rightarrow before(t_2,\ t_4\ )]]]]$$

Clock, time zone and calendar definitions are added to this interval framework to serve as the foundation of the OWL time ontology for the semantic web published by W3C [2]. This time ontology is expressed in OWL DL which, as shown in Section 2.4.4, is a First Order Logic restriction based on $\mathcal{SHOIN}$ DL. This DL is decidable thanks partially to well defined semantics and proven reasoning algorithms. Thus, this qualifies OWL time ontology for time-based reasoning in framework for CPS

design.

Remark 3.1. A number of researchers [18, 78, 185] have determined that interval-based models are more appropriate for formal analysis having time-dependent behavior than other models. In particular, interval-based temporal logics built over Allen's Interval Algebra [18] or Moszkowskis Interval Temporal Logic [185] appear as adequate choices to support practical temporal reasoning for the class of CPS systems of interest. However, it's critical for intervals to be fully specified using the granularity of time implemented in the cyber part of the CPS. Therefore, in this work, we adopt interval calculus approach, which assumes that all intervals are "proper" with a *before* relationship between their beginning and end instants, which are fully specified time points. One of the key benefices of this specification is the avoidance of situations where instances of time intervals are underspecified; that is, either its *beginsAt* and/or *endsAt* properties have no values or the assigned values are underspecified as noticed by Krieger [142]. Moreover, this allows the formulation of restricted axioms which, when expressed in an ontology language, will ensure that time reasoning is decidable.

3.3.4 Comparison of Leading Ontologies of Time

There is currently a wealth of ontologies of time resulting from extensive research in academia and in-house development by industry or joint effort. Even though they seek to formally represent the temporal domain in an unambiguous, human and machine readable way, their features appear clearly different as shown

in Table 3.2. Most state-of-the-art ontologies are developed in OWL with time-interval and/or time-point as theoretical foundations but they define and employ various root concepts. However, most root concepts subsume Time Point and Time Interval which enable the axiomatization of time-interval calculus such as Allen's temporal interval calculus.

When developed as part of foundation or upper level ontologies, the time ontology is merely an extension of such models and is highly influenced by the needs and requirements of those research efforts. It is not the case for stand alone ontologies which requirements and construction are more specifics and tend to be exhaustive. This certainly explains why leading stand alone time ontologies tend to over-perform upper ones when it comes to satisfying the specifications of the reference time ontology (synthetic theory) as shown in Table 3.2. Cyc Time and Suggested Upper Merged Ontology (SUMO) Time appear to be the best foundational ontologies but meet less than 60% of the reference ontology specifications as opposed to OWL Time. The latter emerges as the best stand alone ontology whose 75% of the features match the one of the reference time ontology. Time ontologies are compared on six dimensions as follows [72].

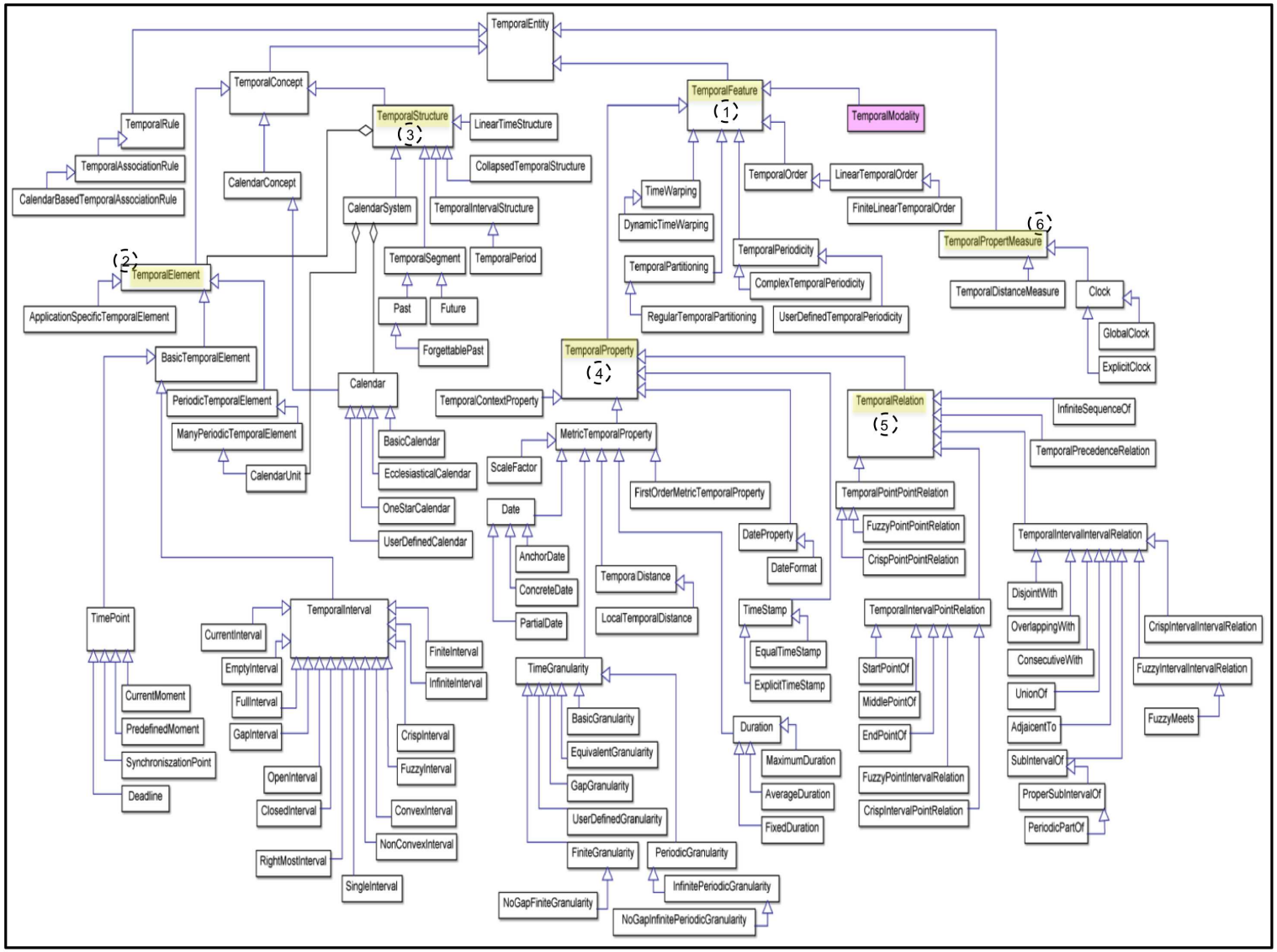


Figure 3.4: The taxonomy of important temporal features as determined by the TIME community [72]. The six concepts whose attributes are used to characterize Time ontology and compare their various implementations are highlighted and numbered. They are chosen based on the importance dimmed by the TIME community of their attributes in characterizing a perfect Time ontology reflecting a " Synthetic theory".

Category	**Time ontology**	**Root concept**	**Theory**	**Implem. Language**	**Degree satisf. synth. theory(%)**							**Description/download#**
					TF	TE	TS	TP	TR	TM	All&	
Foundational or Upper-Level Ontologies	Cyc Time	TimeInterval	Time-interval	CycL	63	29	50	50	100	33	**58**	http://www.cyc.com/platform/opencyc/
	SUMO Time	Time position	Time-point & Interval	SUO-KIF	38	71	0	63	100	67	**58**	http://www.adampease.org/OP/
	DOLCE	Temporal Location	Time-region	OWL	0	7	0	0	0	33	**4**	http://www.loa.istc.cnr.it/old/DOLCE.html
	BFO	Time	Time-region	OWL-DL	19	21	0	6	17	33	**15**	http://jowl.ontologyonline.org/bfo.html
	GFO-BT	Chronoid	Time-interval & Point	OWL	25	57	25	0	58	0	**29**	http://www.onto-med.de/ontologies/gfo/
	PSI-ULO	TimeInterval	Time-interval	OWL-DL	50	57	0	0	0	0	**22**	http://isrg.kit.znu.edu.ua/
Ontologies of Time	OWL-Time	Temporal-Entity	Time-interval	OWL- DL	63	86	63	63	100	67	**74**	http://www.isi.edu/ hobbs/owl-time.html
	TimeLine	TimeLine	Time-interval	RDF	31	57	25	50	33	33	**40**	http://motools.sourceforge.net/timeline/timeline.html
	Reusable Time	NA*	Time-point & Interval	KIF, OKBC	63	93	38	56	100	33	**68**	http://www.ksl.stanford.edu/ontologies/time
	PSI-Time	TimeInstant	Time-Point	OWL- DL	88	86	50	25	100	67	**69**	http://isrg.kit.znu.edu.ua/
	AKT Time	Time-entity	Time-point & Interval	Ontolingua & OCML	13	29	0	50	0	33	**22**	http://projects.kmi.open.ac.uk/akt/ref-onto/
	SWRL Temporal	ValidTime	Time-interval	OWL	0	57	25	63	75	0	**40**	http://swrl.stanford.edu/ontologies/built-ins/3.3/temporal.owl
	SOWL	Temporal-Entity+	Time-interval+	OWL	63	86	50	63	92	67	**71**	http://www.intelligence.tuc.gr/prototypes.php

Table 3.2: Characterization and evaluation of leading ontologies of time.

Legend: * Both Time-Interval & Time-Point were identified as main concepts; +Inferred from OWL-Time given that Temporal concepts in this ontology are defined using OWL-time; & Based on all (36) attributes identified; #Accessed July 22, 2016. Selected parameters and features of the Synthetic theory of time for the evaluation : TF=Temporal Features; TE=Temporal Elements; TP=Temporal Properties; TR=Temporal Relations; TM= Temporal Measures.

• *Temporal features and properties* (TF & TP). Temporal features are primitives that embody the basic theory of time and it incidence (especially in the concept of motion). This translates into the ability of the time ontology to define the right *density*, *order*, *periodicity* of time as well as handling *temporal uncertainty*. Temporal features of the PSI Time ontology appear the closest to the ones required by the reference ontology. Among the critical temporal properties the time ontology should represent are time *granularity*, *scale*, *duration*, *date format* and *timestamps*. OWL Time, SWRL Temporal, SOWL and SUMO Time score the best on this dimension.

• *Temporal Elements and Structures* (TE & TS). Temporal elements are primitive entities that make up a temporal theory as introduced in the taxonomy in Figure 3.4. Thus, the time ontology should be able to properly define and capture temporal primitives such as *Time Points*, *Temporal Intervals* or *Temporal Segments* . Almost all stand alone ontologies perform well on this dimension with the Reusable Time being the best. On the other hand, Temporal structures are purpose-driven compound constructs built from base primitive structures and temporal elements such as *temporal periods* or *Calendar*. This seems to be a hard to meet specification as almost all ontologies fail to get past the bar of 50% match of the reference ontology. However, OWL Time ontology stands among all the ontologies considered.

• *Temporal Relations and Measures* (TR & TM). Relations between temporal entities and structures are binary properties linking them. Even though the arity of the relations can be higher than 2, relevant ones do not exceed that limit. Thus, the ontology should be able to represent *Interval-to-Interval* (crisp and/or fuzzy), *Interval-to-Point* and *Point-to-Point* relations. Many upper (e.g.: Cyc, SUMO)

and stand alone (e.g.: OWL, Reusable Time, PSI-Time) ontologies fully satisfy the expectations of the reference ontology.

Figure 3.5 summarizes the comparison between the leading upper and standalone time ontologies. None of them matches all the specifications of the reference ontology as per the synthetic theory, nor dominates all the others ontologies in all the dimensions. However, OWL Time appears to match or exceed the performance of the others in all the dimensions except for the Temporal features where it's dominated by PSI-Time.

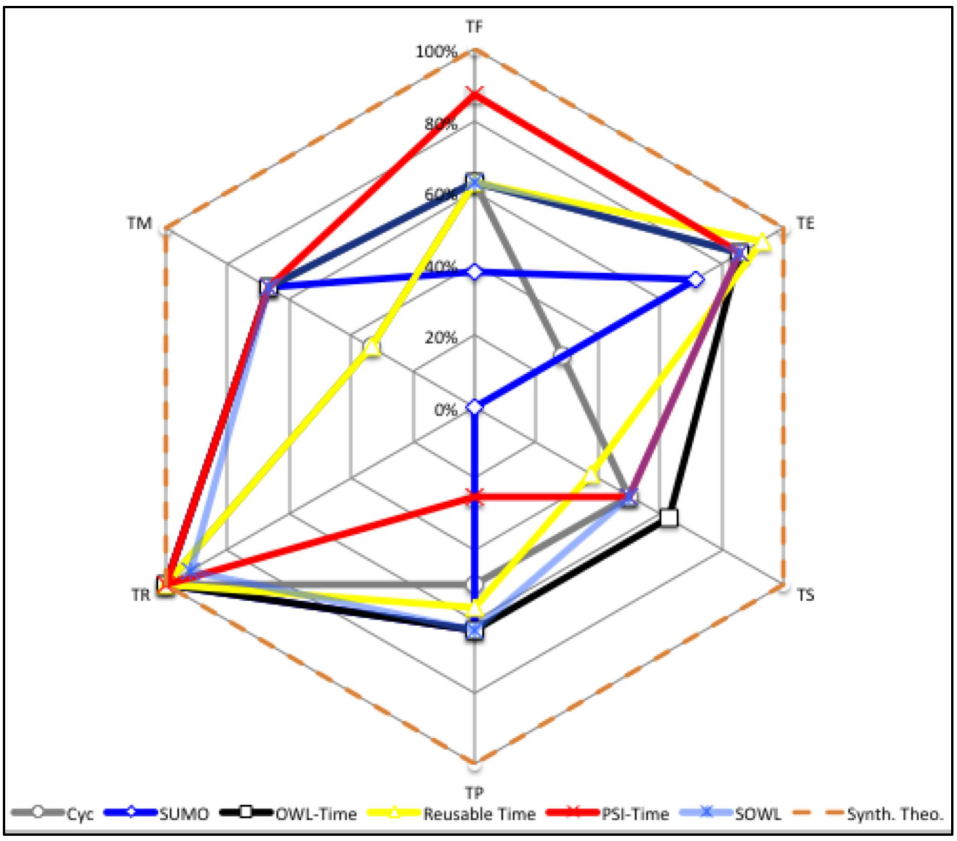


Figure 3.5: Comparison of leading ontologies of time to the reference ontology (Synthetic theory) as defined by the TIME Community. The six dimensions of the evaluation are as follows. TF=Temporal Features; TE=Temporal Elements; TP=Temporal Properties; TR=Temporal Relations; TM= Temporal Measures.

3.4 Temporal Reasoning and Rules

3.4.1 Temporal Logic

Temporal logic plays an important role in systems design when we want not only to know what is true, but when? Mathematical formalisms for temporal logic are particularly useful for describing the properties of concurrent systems, where individual processes must be coordinated in order for "correct behavior" to occur [231]. In this context, behavior means how a system will react to external stimuli and internal events (critical to reactive systems and real-time systems). Constraints tend to fall into two categories: 1. Events and event orderings, and 2. Quantitative temporal constraints.

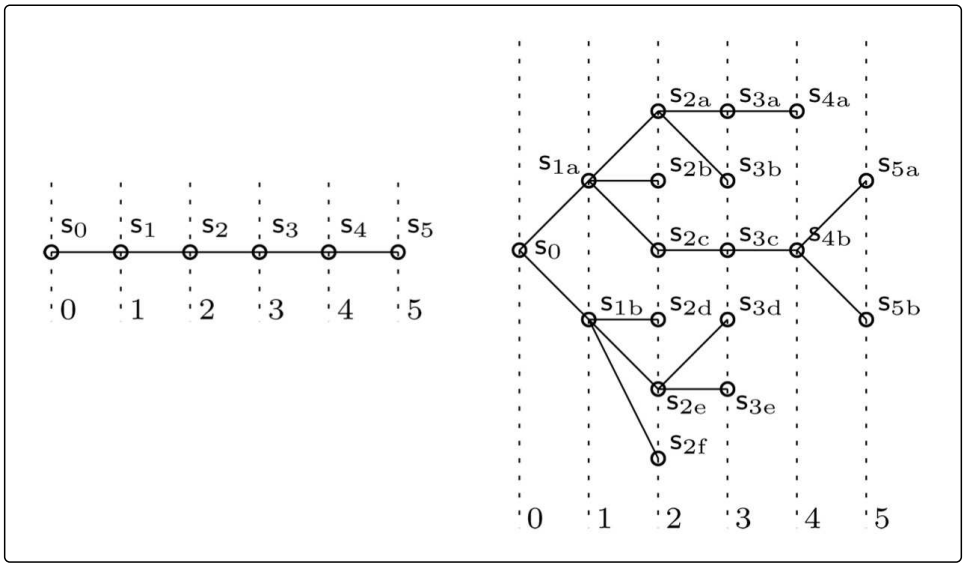


Figure 3.6: Schematic for linear (left) and branching (right) temporal logic.

The terms linear and branching refer to the structures upon which a formal logic is interpreted. In both cases, linear and branching time logic, system behavior is

described in terms of actions and state sequences. As illustrated on the left-hand side of Figure 3.6, A model of linear-time temporal logic (LTL) is an infinite linear sequence of states. where each point in time has a unique successor. Temporal formulas are evaluated over such a sequence of states together with an index i=0,1,2,... of the i'th state. A model of branching temporal logic (BTL) is an infinite sequence of states where each point in time may have multiple successors. Branching time describes all possible time lines.

3.4.2 Jena Rules for Temporal Reasoning

Rules are the underlying mechanisms and enablers of inferencing services introduced in Section 2.2.2. In Jena, multiple inference engines or reasoners can be configured and plugged to allow the derivation of additional RDF assertions. Rules are generally of the form "*If...(condition)... then, ... (consequence)* " statements (more in Section 5.3.5). The first part (or body) states the set of conditions that should be satisfied before the rule is fired while the second part (or head) specifies the new state. Rules formalize relationships and interactions between entities types in a domain, as such they apply to the *TBox* of the knowledge base $\mathcal{K}\langle\mathcal{T},\mathcal{A}\rangle$ as introduced in Section 2.2.2.

Jena enables the formulation of rules to characterize a certain number of properties and perform qualitative and quantitative evaluations on a domain. In the case of temporal domain, let's consider three rules for the following purposes : (R1) to characterize the order of occurrence between time instants, (R2) to compute the

duration of time intervals or, (R3) to specify the "intFinishes" relationship between two intervals as per ATIC. The following excerpt shows the Jena implementation of the selected rules.

```
// Rule 1: Deduction of happensBefore relation between time instants...

[ HappensBefore: (?x rdf:type af:Instant) (?y rdf:type af:Instant)
  (?x af:hasTime ?t1) (?y af:hasTime ?t2)
  lessThan(?t1,?t2) -> (?x af:happensBefore ?y) ]

// Rule 2: Compute and store the duration of a time interval...

[ GetDuration: (?x rdf:type af:BeginEndTimeInterval) (?x af:beginsAt ?y)
  (?x af:endsAt ?z) getDurationInterval(?y,?z,?d) -> (?x af:hasDuration ?d) ]

// Rule 3: Deduction of intFinishes relation between time intervals...

[ IntFinishesRule: (?x rdf:type af:ProperTimeInterval)
  (?y rdf:type af:ProperTimeInterval) (?x af:endsAt ?t)
  (?y af:endsAt ?t) -> (?x af:intFinishes ?y) ]
```

3.5 Case Study: Temporal Modeling and Reasoning in Action

Figure 3.7 illustrates a case study problem of modeling in the temporal domain with ontologies and rules.

3.5.1 The Time Ontology

The upper right-hand side of the figure shows an excerpt of the time ontology with a representation of the relationship among classes and properties. A temporal entity has properties: `#beginsAt`, `#endsAt`, and `#hasDuration`. `#Interval` and `#Instant` are subclasses (specializations) of the class `#TemporalEntity`. `#ProperTimeInterval` and `#OpenTimeInterval` (not shown) are specializations of class `Interval`. An

#Instant is a #TemporalEntity that has a clock time attached/associated to it. The following excerpt illustrates the formal description of a #ProperTimeInterval.

```
<owl:Class
    rdf:about="http://www.isi.edu/~pan/damltime/time-entry.owl#ProperTimeInterval">
    <rdfs:subClassOf rdf:resource="&time-entry;TemporalEntity"/>
    <owl:disjointWith rdf:resource="&time-entry;Instant"/>
    <owl:disjointWith rdf:resource="&time-entry;OpenTimeInterval"/>
</owl:Class>
```

Properties can be distinguished by their domain and range. Both the domain and range of ObjectProperty are ontological classes. Restrictions (e.g.: Reflexive, Symmetric, etc.) can be added to this type of property to constraint their behavior and their inverse can also be defined. Unlike objectProperties, the range of DatatypeProperty in ontologies are primitives or specialized datatypes. For instance, the following excerpt describe the #hasTime DataProperty. Its range is an entity of type *dateTime*

```
<owl:DatatypeProperty
   rdf:about="http://www.isi.edu/~pan/damltime/time-entry.owl#hasTime">
   <rdfs:domain rdf:resource="&time-entry;Instant"/>
   <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>
```

3.5.2 Semantic Graph Transformations

The upper left-hand side of Figure 3.7 shows a set of facts and three rules. The facts are: (F1) An entity E1 is at location X at time tX and at time tB at location B. (F2) Entity E2 is at location S at tS and at B at tB. (F3) Interval tXB starts and ends at the same time respectively as tX and tB. (F4) Interval tSB starts and

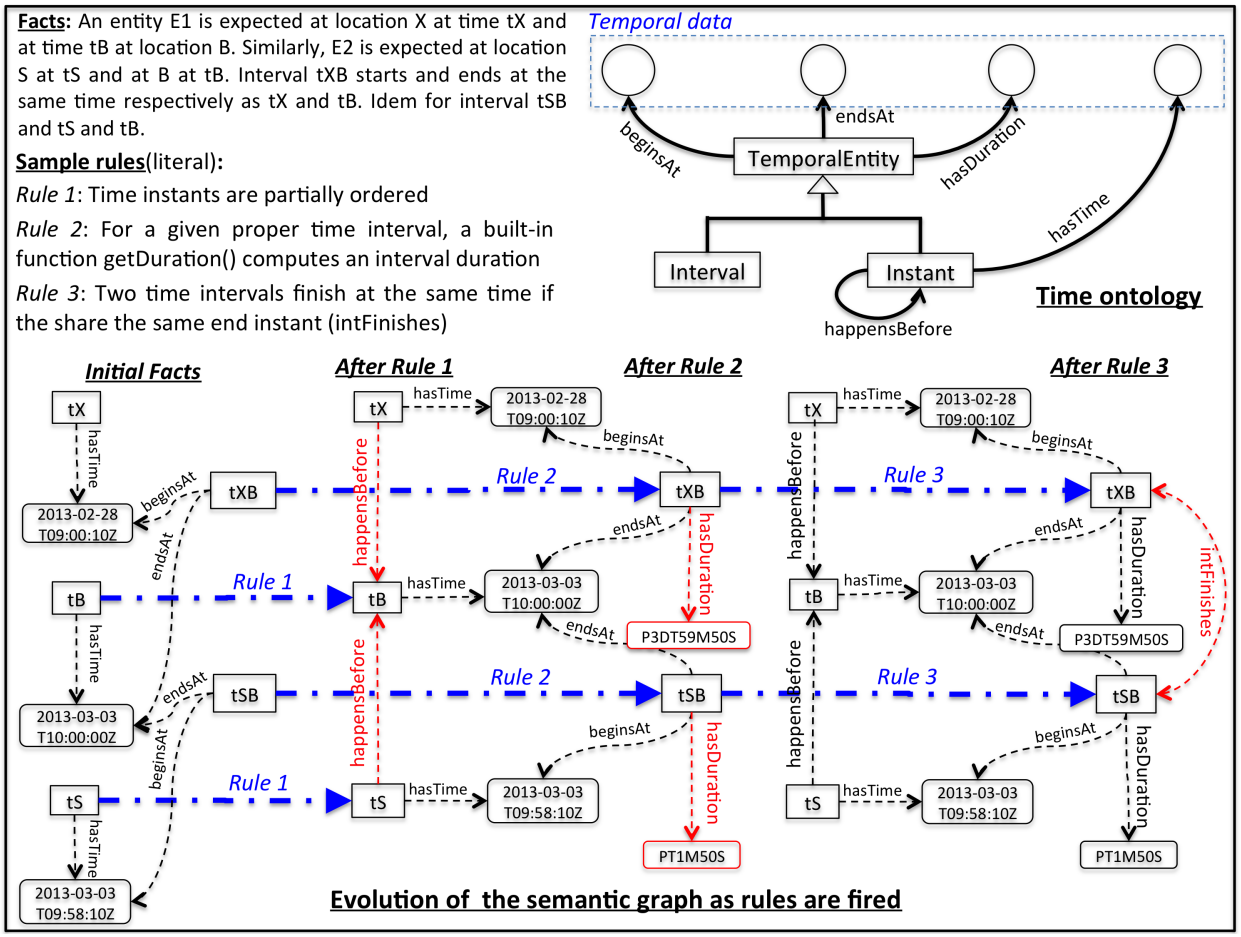


Figure 3.7: Illustration of semantic-driven modeling and reasoning in the temporal domain.

ends at the same time respectively as `tS` and `tB`. The rules are (R1)-(R3), defined in Section 3.4.2. The rules are applied to the ontology to transform the structure of the time semantic graph through inferencing as introduced in Section 2.2.2. First, the ontology is initialized with the set of facts (F1)-(F4). To that aim, the capability of the Jena API is used to extract classes and properties from the loaded ontology, then create semantic-compliant temporal data and finally add the fact as a statement in model of the semantic graph for the temporal domain. As an illustration, the key steps to encode the first part of fact (F1) are as follows.

```
// Create individual tX as an instance of the class TemporalEntity ....

   tX = modelTime.getOntClass(ns + "TemporalEntity").createIndividual(ns+"timeAtX");

// Extract hasTime property from the loaded time ontology; add to ‘‘modelTime"....

   hasTime = modelTime.getDatatypeProperty(ns + "hasTime");

// Encode the temporal data using the valid dateTime representation

   Literal timeX = modelTime.createTypedLiteral("2013-02-28T09:00:10Z",
                                                XSDDatatype.XSDdateTime );

// Write the fact as a statement in the form Subject-Predicate-Object

   Statement timeAtXinstant = modelTime.createStatement( tX, hasTime, timeX );

// Add the newly formed statement to the semantic graph

   modelTime.add ( timeAtXinstant );
```

The schematic along the bottom of Figure 3.7 shows the evolution of part of the time semantic graph as the three rules (R1)-(R3) are executed as a function of time. The first view (Initial Facts) is a representation of the graph after all the facts (F1)-(F4) are encoded. Some of the data such as the type/class of individual temporal entities (not shown) remains constant over time. Other data and relations between entities are dynamic and controlled by time domain rules. As a case in point, after rule R1 is executed, the graph is expanded with creation of the `#happensBefore` objectProperty between `tX` and `tB` on one hand and `tS` and `tB` on the other hand. rule R2 adds new data (the duration of time intervals) and property (`#hasDuration`), which further expand the graph. The last rule creates the symmetric `#intFinishes` objectProperty between intervals `#tXB` and `#tSB`.

Figure 3.8 shows some of the statements on the time interval tXB stored in the data repository after all encoded domain rules are executed on the initial set of

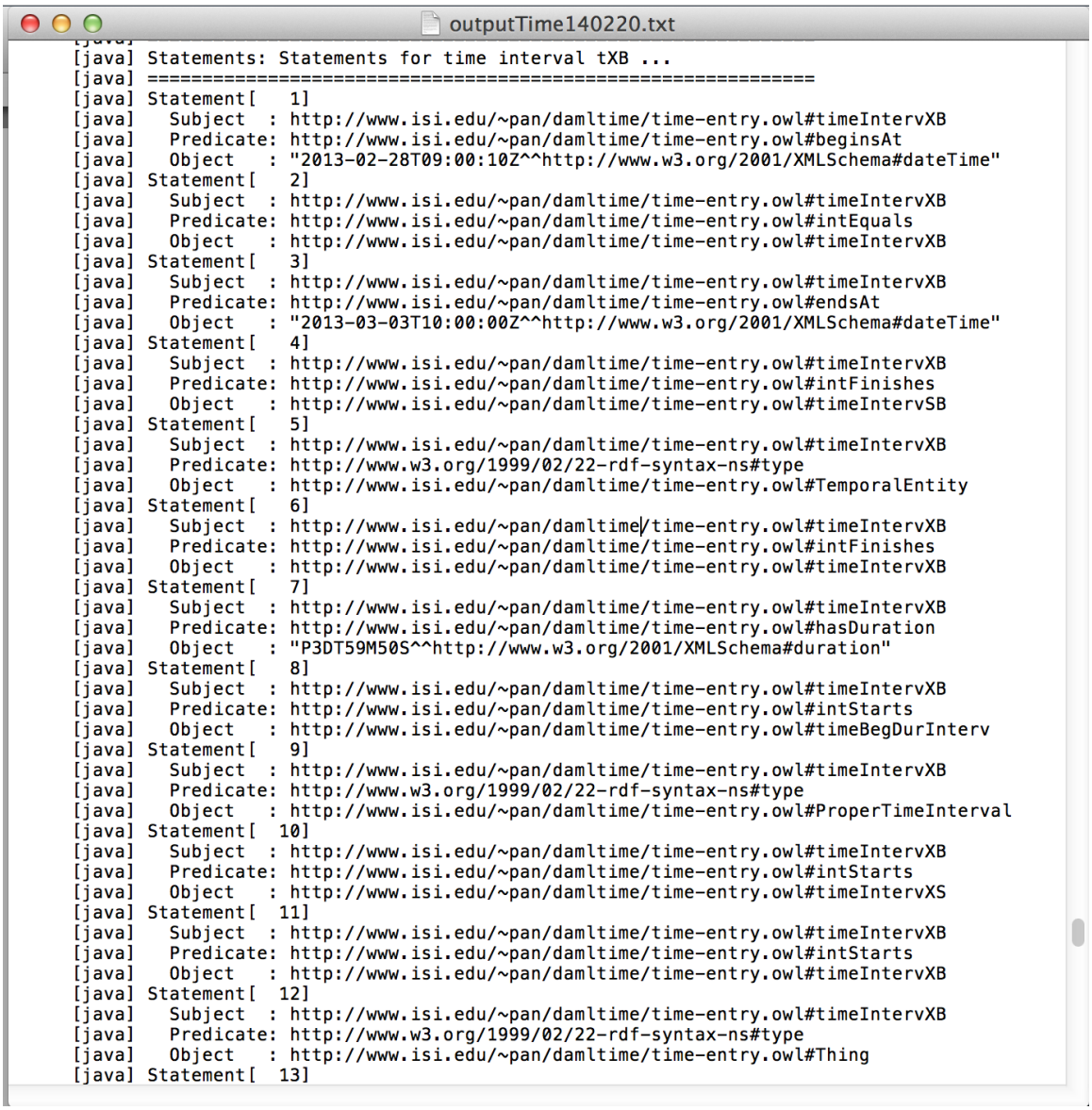


Figure 3.8: Excerpt of statements relative to the time interval tXB stored in the data repository after all encoded domain rules are executed on the initial set of facts. Initial facts regarding tXB is translated by statement #1 and #3 (for F3). The impact of the selected set of rules is visible in statements #7 (for rule R2) and statement #4 (for rule R3). Rule R1 does not have any effect on tXB.

facts. The initial facts regarding tXB is translated by statement #1 and #3 (for F3). The impact of the selected set of rules is visible in statements #7 (for rule R2) and statement #4 (for rule R3). Rule R1 does not have any effect on tXB. We also note multiple other statements such as #2, #8 to #13 resulting from other rules.

Remark 3.2. At execution time, rules are not fired in a particular order but they are not fired in a complete random manner neither. The statements in the data repository are constantly evaluated with respect to the premise or body of rules before the rule is fired. However, the modeler can add control mechanisms to control/enforce a certain order in the execution of the rules. Also, Jena provides mechanisms to reduce the size of the graph by removing statements. The latter can be replaced by new ones when updating the graph without expanding its size.

Chapter 4: Semantic Modeling of Space

4.1 Introduction

This chapter examines spatial semantics and their use in supporting the creation of accurate, precise, scalable and reusable models of space in the context of safety-critical cyber-physical systems (CPS) design. The central premise of our work is that ensuring the safety of such systems requires the development of a scalable, flexible, and customizable ontological framework that supports the embedding of physical semantics into cyber models for system smartness. Thus, we need ambiguity-free models of space that properly capture the spatial configuration of the system as it's materialized in the world. This is an essential foundation for reasoning tasks involving spatial entities.

We discuss the key role that ontologies can play in capturing and formally representing the space domain. Spatial theories and semantics supporting the formalization of spatial knowledge and the decidability of derived spatial reasoning systems are reviewed in Section 4.2. The ontological descriptions of this domain are conducted in Section 4.3. In Section 4.4, we develop and describe a simple,

multi-dimensional tree structure of spatial models that support the representation of spatial entities at various level of granularity and enable the use of associated operations and predicates essential for reasoning using complex spatial datatypes. We highlight the central role of the region connectedness calculus (RCC-8) algebra and spatial relationships to support the reasoning about space and spatial regions. The supportive geometry algorithm for the implementation in Java is introduced in Section 4.5. A case study demonstrates the use of semantic web technologies to support spatial knowledge representation and reasoning is described in Section 4.6.

4.2 Space and Spatio-Temporal Theories

4.2.1 Spatial Theories and Calculus

Theories of Space. As for time, there is a need for formal definition of space to support the ontological modeling of this domain and systems in which they play an important role, especially when it comes to safety. Thus, we revisit Vieu's views in [276] and we adopt a mereotopological categorization of spatial theories which mirrors - to a certain extent - the one of temporal theories. However, unlike time, space is not oriented, nor cyclic. This leads to the following classification of main spatial theories and calculus.

i. **Space-point.** Space is perceived as arrangement of points with focus on orientation and distance concepts. Other extended spatial entities such as lines and regions are defined as sets of points. This is the view adopted in mathematical

theories of space [118, 235, 275]. However, in [39], points are centers of regular 3D shapes (sphere).

ii. **Space-interval.** Tuples of intervals resulting from the projection of regular regions (i.e., rectangular shapes) on the axes of a reference frame are the primitive spatial entities in this class of spatial theories. These theories are mostly inspired and attempt to mimic Allen's temporal calculus [110, 188]. However, they go beyond mereotopological information to account for orientation information as well.

iii. **Space-array.** In this theory, space is a collection of arrays, i.e., a discrete coordinate system. It has the advantage to concurrently capture topological, orientation and distance information all at the same time. This theory is widely use to support computer visualization and spatial databases applications as well as linking visual and linguistic spaces [96, 111, 150].

iv. **Space-region.** A region of any shape with dimension higher than one is the primitive in these theories. However, regions should be of the same dimension as the whole space and their interior should not be empty. Major variations include the earlier version of the region-based theory axiomatized around the connection relation C and further extensions [51, 274, 282]. Also, theories built from one mereological (part) and one topological (contact or external connection) relations belong to this category [39].

v. **Space-multidimension.** There is no restriction on the dimensionality of spatial primitives nor on the one of the whole space in these theories. They do not assume nor define a hierarchy between their primitives, but introduce incidence relationship in lieu of ontological dependency. Some of theses theories focus on rendering multiple dimensions as do humans [89, 101] while others introduce and support the notion of boundary [247]. Plus, the mathematical expressivity of such multidimensional spatial theories is of interest for some researchers [53].

Additional Considerations. Despite the high variety and depth of spatial theories, a full accounting of space remains more challenging than one of time. Theorizing space is rendered more complex because of non-mereotopological aspects such as *dimension, orientation, shape, length, area* or *volume* that are relevant in many applications such as robotic or engineering design. Existing theories accounting for these aspects often involve explicit triadic relations as it's the case of (1) Cyclic order (CYCORD) relation-based calculus [227] or points-qualitative values function mapping calculus for orientation [235] and, (2) CanConnect primitive-based calculus [64] or delta calculus [302] for distance and size. When it comes to *shape*, various approaches based respectively on slope projection, curvature and boundary segments [137, 165, 226] are among those that have been investigated. Also, *latitude, longitude, elevation, geopolitical subdivisions or aggregates* are of high importance in Geographic Information systems (GIS) applications [120].

On the other hand, results of the *composition of spatial primitives* from mereological and topological representations are not always unique. Moreover, in spatial systems, the accuracy of computation and control often depends on the number and location of sensors as well as their capabilities. If the sensors are moving, then timeliness of computations will be affected by the velocities of both the sensors and objects moving throughout the environment.

Selecting a Spatial Theory. The complexity involved in the formal description of space does not make the selection of a given theory an easy task. However, the presence of physical entities in complex systems such as CPS dictates that one stays away from pure philosophical debates - such as the existence or not of vacuum [16, 287] - in the selection approach. Thus, theories that foster geometrical or physical structure representations of space while enabling the addition of key extensions above, will be considered for their practical ability in supporting reasoning tasks. This is consistent with the Newtonian view of space which distinguishes space from the objects with a location within it as opposed to Leibnizian approach which defines space in term of inter-relationships between objects [39].

Remark 4.1. *(Uncertainties in Spatial Knowledge Representation).* As already seen for time, notions of space can be under-specified in utterances and natural language expressions. For instance, consider the following expressions: (a) The car wandered *around* the 188 train accident scene and, (b) The car wandered *to* the 188 train accident scene. As pointed out by Thorton [267], the pairing of the non-directional verb "wander" and the prepositions "around" and "to" leads to

ambiguity that needs either resolution or semantic coercion to properly interpret the notion of place (a) and path (b). Furthermore, the dimension(s) of the two spatial concepts (i.e., place and path) is (are) unknown, but it (they) can be very relevant as we'll later see.

Also, traditional representation approaches of spatial knowledge generally assume that (1) boundaries of spatial regions are well-defined and, (2) regions can be physically observed and rendered as sharp objects. However, Freska [83] points out that, this assumption is inappropriate since real world limitations in knowledge acquisition makes uncertainty inherent to the spatial data captured and represented.

4.2.2 Spatio-Temporal Theories

Space-Time. Given the prospect of increasing complexity in handling separately temporal and spatial theories within a common reasoning framework researchers have looked into ways to formalize space and time into single space-time theories. This adds in complexity to the challenge of ensuring satisfiability of the reasoning process for system models that rely on such theories. The satisfiability of the reasoning is now conditioned by the actual existence, in the real world, of the logical configuration inferred. Thus, the world of the system (such as a CPS) should be a "living" one, where space and time are clearly defined and well understood. In [265], the author postulates that, in such a world, the dimensionality of space-time must be (3+1). He shows that the hyperbolicity property that enables observers to make predictions will lack in partial differential equations if the dimension of time is dif-

ferent from 1. Also, space with less than three dimensions won't contain observers, nor allow gravitational forces. Plus, the stability of structures will be problematic and the fundamental existence of traditional atoms will be questioned in space with more than three dimensions.

Spatio-Temporal Theories. One of the main interests in plain spatio-temporal theories lies in the ability to formally describe and reason about motion in a qualitative way. Also, changes in spatial entities over time is another important need, especially in GIS [98]. These capabilities are highly influenced by the foundational view of the universe adopted by the developer of the theory. For this research, those theories that are flat space-time geometry - as per either Minkowsky view (space-time as whole) [182] or Newtonian (space + time independent) view - appear to be the most attractive.

Proponents of unification have focused their effort on developing theories that attempt to construct a unified representation from foundational temporal and spatial theories introduced above. Some have centered their theories on topological aspects of space-time with various primitives such as: (a) space-time histories [187], (b) temporal space [49] and, (c) spatio-temporal trajectory (STT) [296]. Another unified theory constructs spatio-bitemporal objects (ST-simplex and ST-complex) from 2D spatial and temporal primitives [289]. However, because of the reliance on 2D temporal entities, this theory does not satisfy the hyperbolicity property, thus, it's inappropriate in the context of this research.

Researchers that make use of Newtonian view of the universe maintain a

clear separation between original spatial and temporal theories and rely on various solutions to address the challenge of linking time and space in describing motion or events. One possibility explored by researchers has been to borrow concepts and ideas from graph theory to support semantics of graph-based models for spatio-temporal evolution [63, 93]. Another direction investigated has been the definition of hierarchical relationships in temporal and spatial domains. This effort has been coupled with the development of modular integration procedures that enable both the location of events in space and spatio-temporal queries [50].

4.3 Ontological Descriptions of Space

4.3.1 Ontologies of Space

The increasing need for unambiguous and formal qualitative account of space, location and movement in space in various areas such as robotic [144], urban environments [151], science [236], geographic information systems [81,217] has driven the development of spatial ontologies in those domains. Research in some of these areas has led to the development of domain specific ontological languages. For instance, the geographical markup language(GML) [11] has been proven effective in representing geo-ontologies especially in the context of retrieval of geographic information on the web [13]. Similarly, CityGML has been used to represent 3D urban objects in city models [107].

Bateman et al. [30] try to break away from application domain dependence

of spatial ontologies by introducing"detailed semantics" for linguistic spatial expressions supportive of computational processing that draws substantially on the principles and tools of ontological engineering and formal ontology. Because of the narrow view adopted by many of these "domain oriented approaches" and their ontological considerations, resulting ontologies of space are limited in scope, access and uses. However, independently of their applications, ontologies of space need to support models of space that are three-dimensions (or less) and work with ontologies and models of time that are one dimensional. Thus, we ought to investigate ontological approaches that embrace a broad view of space i.e. 3D upper ontologies that guarantee the satisfaction of the hyperbolicity property (see Section 4.2.2) when combined with 1D time in space-time.

4.3.2 Classes of Spatial Ontologies

Spatial ontologies can be organized into hierarchies of spatial concepts (a taxonomy), and can be made more rigorous through the addition of axioms [6, 179]. They can be grouped into two main categories as follows:

1. *Hierarchical Spatial Ontologies.* In this class of ontologies, entities definitions are classified and organized hierarchically, resulting in a tree structure. In SUMO [6], spatial entities are part of the physical universe, thus, its use of a "physical concept" as the root of the tree as shown in Figure 4.1. Objects in SUMO-space are defined with respect to their shape and position as attributes. A "SpatialRelation" class generalizes a taxonomy of relationships between

spatial entities. OpenCyc [7] follows the same decomposition principle as SUMO, but it uses a different root i.e. "SpatialThing". However, the latter does not make a formal reference to objects of any kind. The reference is made to concrete, observable spatial categories through the notion of "SpatialThing-Localized." OpenCyc has an extensive library of path constructs that make up its path system. In this category, we also list the dormant effort of developing OWL-space and linking it with SUMO and OpenCyc. It has been viewed as a counterpart of OWL-time and is been developed as an extension of the DAML for space with GIS as the primary application area [120].

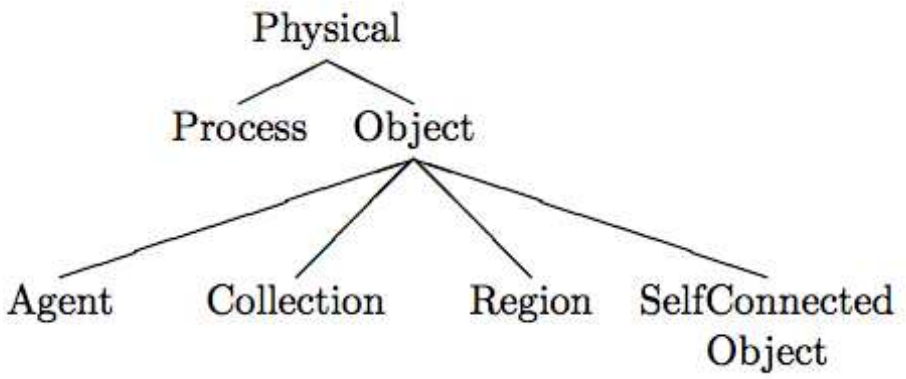


Figure 4.1: Taxonomy concerning physical in SUMO [31]

2. *Axiomatized Spatial Ontologies.* This category of ontologies is characterized by rigorous ontological definitions strengthened by axioms. The spatial dimension of DOLCE [179] has an intentional bias toward cognitive knowledge to make it more suitable for use in the context of semantic web. Thus, "entity" is the root element of the taxonomy whose categories are rigid properties. Also, DOLCE adopts a dynamical view of spatial entities by distinguishing between enduring

and perduring entities. Axioms in basic formal ontology (BFO) [248] are built under the premises that reality can be described using two kinds of ontologies: SNAP and SPAN ontologies. The former is purely spatial and provides a snapshot of the reality though a description and relationships between constitutive endurant elements. Time is intrinsically present in the latter, thus, entities in SPAN ontologies are located in a space-time continuum. They unfold themselves with respect to a given time interval, domain of reality, and at some spatio-temporal level of granularity. Overall, DOLCE can be considered as a special type of SPAN ontologies while SUMO and OpenCyc would be classified as SNAP ontologies.

4.4 Multi-Scale Spatial Modeling and Reasoning

4.4.1 Space Matters: Need for Formal Models of Space for CPS

Space Matters. In order for formal approaches (such as model checking and theorem proving) to the verification of CPS to be effective, system models need to capture the appropriate granularity of space considering it can be under-specified as shown in Section 4.2.1. State-of-the-art models of safety-critical systems and formal verifiers use 0D models of space [95, 259] built on space-point theories.The absence of spatial boundaries in these system models makes it impossible to properly track the interactions between the system elements, especially when they are software-intensive and distributed as in most CPS [162]. In fact, among the five types of spatial theories listed in Section 4.2.1, none effectively captures both the

mereotopological and non-mereotopological aspects of space for CPS modeling in a practical manner.

Spatial Models. Spatial models can be classified as being either symbolic or geometric [15]. Geometric models make use of cells and/or boundaries as primitives model entities. Symbolic models use topological-based structures and/or graphs to capture connectivity, reachability and hierarchies between spatial entities. Even though the latter class of models provides semantically compliant entities location (partially) in a human-readable way along with topological relationships, their ontological commitment with regard to the spatial theories introduced in Section 4.2.1 is ambiguous. Thus, they can't be systematically traced to a sound logical foundation. This makes their use in the context of reasoning for safety-critical CPS applications inappropriate.

This observation points to a strong need for geometric models, with primitives specified at the desired granularity of space. Fortunately, for the family of CPS considered in this research, the constraints of determinism and precision on models can be translated, with regard to the account of space, into mereotopological-related descriptions. Therefore, theories with strong mereotopological focus and mechanisms to account for some relevant non-topological aspects are acceptable spatial foundations for our models. To that extent, space-region theories appear to be excellent candidates, thus our choice of the Region Connectedness Calculus (RCC) [56, 225, 233].

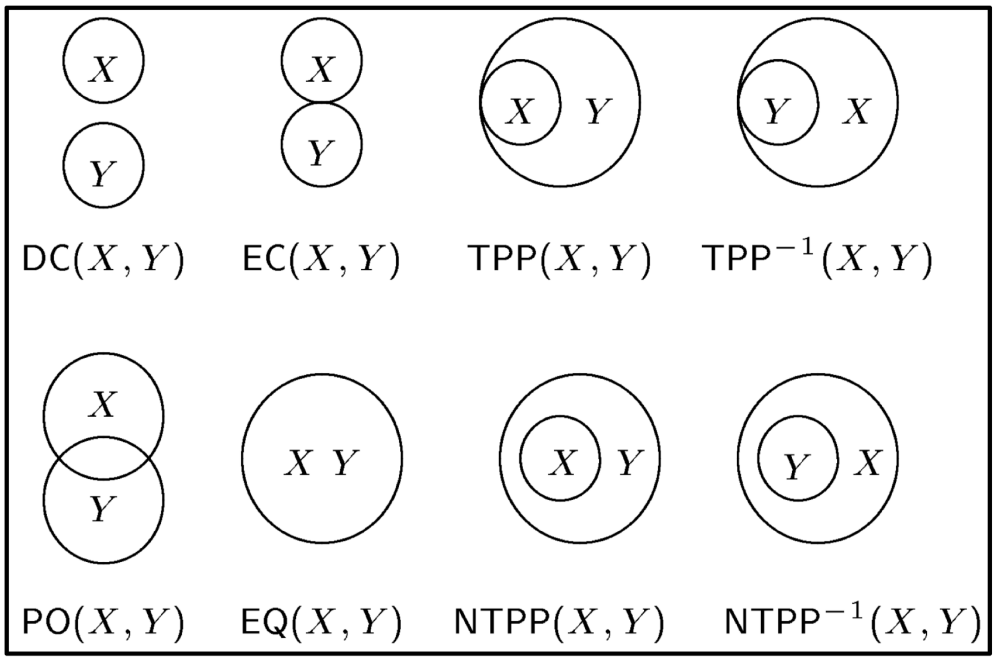


Figure 4.2: Eight types of relationships between spatial entities in Region Connection Calculus (RCC-8).

4.4.2 Qualified Theory: Region Connectedness Calculus

Overview. The region connectedness calculus is a space-region theory for space. Its beauty lies in its strong mereotopological focus and flexibility to seamlessly integrate with "low dimension" theories and extensions to account for key relevant non-topological aspects such as distance, area, volume and other relevant features. Also, this spatial calculus is precise enough to clearly distinguish convex to concave shapes and it can handle uncertainties in regions' boundaries. Moreover, it provides efficient support to inferencing in static and dynamic situations, a capability critical for qualitative reasoning about motion.

At the core of this algebra is the relationship between spatial regions. Given two spatial regions *S1* and *S2*, a space-point p and a proposition ϕ, we might ask a

variety of questions over the space domain such as: (1) **Mereological** or part-of questions (e.g., Is the region *S1* a subset of *S2* ? Does p lie within *S1* ? Is the region *S1* equals to *S2* ? (2) **Topological** or "connects" questions (e.g., Do intervals *S1* and *S2* meet ? Do regions *S1* and *S2* overlap ?) and, (3) **Logical** or rules-based questions (e.g., Does the proposition ϕ hold within the region *S1* ? If ϕ holds within the region *S1* does it hold within *S2* too ?)

Cohn [56] has identified and specified eight (8) relationships - based on the primitive relation "connection" C - between any pair of regions as the core of this Algebra, thus the name RCC-8(there is a RCC-5 version too). Those relationships are illustrated in Figure 4.2. The excerpt below illustrates the definition of *Part*, *Overlaps* and *Partially Overlaps* relationships between 2 given regions x and y.

P(x,y) : $\forall z[C(z,x) \rightarrow C(z,y)]$; *x is a part of y*

O(x,y) : $\exists z[P(z,x) \wedge P(z,y)]$; *x overlaps y*

PO(x,y) : $O(x,y) \wedge \neg P(x,y) \wedge \neg P(y,x)$; *x partially overlaps y*

Restrictions. One limitation of RCC is its inability to make a clear distinction between open and closed regions as well as the dimension of spatial regions. On the other hand, results of the composition of spatial primitives from mereological and topological representations can result to multiple possible spatial configurations in the world which can not be properly captured by the reasoner. Thus, we need to add restrictions to RCC models with the primary concern of ensuring decidability of spatial reasoning. These restrictions include, but are not limited to, spatial entities with shape as regular as possible and limitation to pair-wise (mereo)topological

relationships. Also, in order to maintain the hyperbolicity property for space-time interactions, we restrict the maximum dimension of space to three (3), which does not eliminate the possibility to navigate to and visualize lower dimension spatial entities. These constraints allow the formulation of restricted axioms which, when expressed in an ontology language, will ensure that spatial reasoning is decidable.

4.4.3 Spatial Modeling Architecture and Description

In this section, we introduce a new spatial-based modeling and reasoning architecture to support the modeling of space, as a metadomain in the context of safety-critical CPS design. The system architecture is shown on the left-hand side of Figure 4.4.

1. Multidimensional Spatial Modeling. This module provides to others the formal model of space in conformance to the spatial theory of interest i.e. restricted RCC-8 in this case. Model entities are organized into an hierarchy of four types of spatial entities enriching each other from top to bottom as shown in Figure 4.3. However, given that each type of model is from a different dimension, they can each stand by themselves while enabling the representation of spatial entities at various levels of fidelity using OD (point), 1D (line), 2D (polygon) and 3D (polyhedra) representations as shown in the middle of the figure. For each of these representations, a specific type of geometry will ultimately support the encoding and storage of spatial data of the entity subject to analysis and reasoning. A given layer of the hierarchy is typically composed of three types of spatial entities as follows.

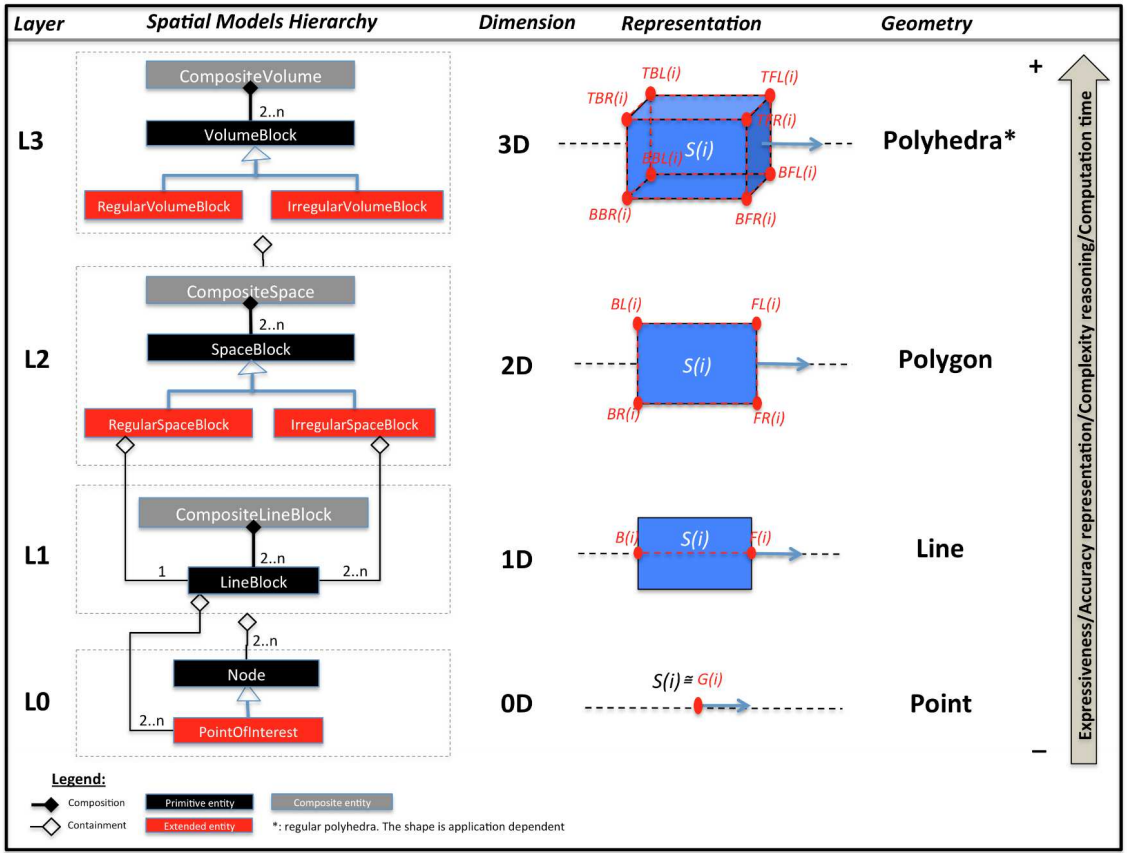


Figure 4.3: Spatial models hierarchy and representations.

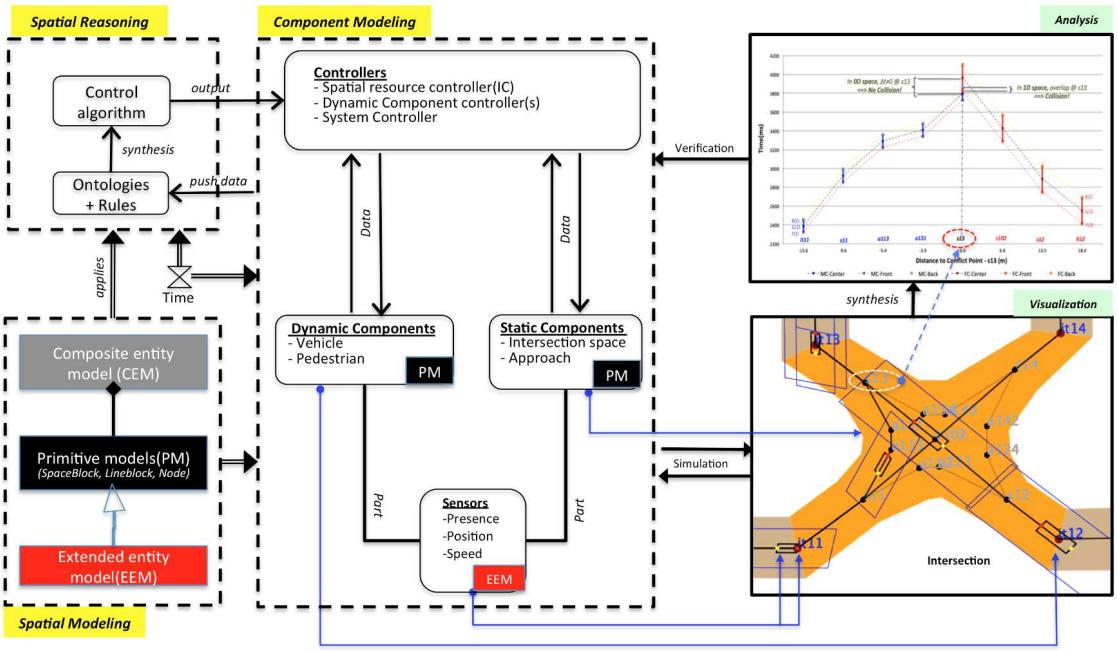


Figure 4.4: Spatial modeling and reasoning framework extended with views for race track simulation and analysis. The three main elements of the framework are: (1) models of space, (2) models of components, and (3) support for spatial reasoning.

a/ Primitive entity. This is the foundational model of space for the dimension considered. It directly emulates the foundational primitive concept in the restricted spatial theory. Thus, Node, Lineblock, Spaceblock and Volumeblock are respectively translations of point (space-point), and regions in dimensions 1, 2 and 3 in RCC-8.

b/ Extended entity. It's an enriched version of the primitive entity with additional non-mereotopological attributes and features that may be particularly relevant for the application of interest. This entity also offers ways to differentiate between model entities of the same dimensions, as seen for 2D and 3D entities.

c/ Composite entity. Composite entities are made of the composition of two or more primitive (or extended) entities within the same dimension. The "composition" of spatial entities at a given level implies the composition of lower level entities, if they are part of the top level entity.

The "containment" connector is a weaker "composition" between spatial entities of higher and lower dimensions. It helps define and refine the definition of spatial entities at various level of the hierarchy. Also, the arrow on the right side of Figure 4.3 shows that the expressiveness and accuracy of the spatial model come at a cost of higher complexity and computation time. Moreover, in spatial systems, the accuracy of computation and control often depends on the number and location of sensors as well as their capabilities. If the sensors are moving, then timeliness of computations will be affected by the velocities of both the sensors and objects moving throughout the environment.

2. Component Modeling. In the context of CPS modeling, spatial model en-

tities do not stand by themselves. They are enrichment and properties of objects and components in the real world. For instance, a "Vehicle" object can be defined by the properties model, make, usage, maximum speed and owner. Adding the positional information on its geographical location such as its (x,y,z) geo-coordinates turns the vehicle into a spatial object. The decision to "spatialize" components of the CPS is dictated by the purpose of the application, the targeted analyzes and the role they play in the system. Such components are marked with the stamp of the corresponding spatial entity extension as shown by the PM and EEM annotations in the central part of Figure 4.4. In safety-critical applications, we can differentiate dynamic components (those whose location evolves with time) from static ones (those that do not). Sensors are mounted on components (mobile or not) and they have extended entity spatial model stamp. Actuators are left out this component model, however, they can be added as part of the component in a way similar to sensors.

3. Spatial Reasoning. Reasoning occurs at various levels of CPS in support of system control (locally and globally). Thus, both control algorithms and reasoners are an integral part of reasoning in the proposed framework. Irrespective of where it occurs, reasoning involves the inputs, i.e., data from the component module, the construction of facts and inferencing of new facts that are synthesized by the controller using the appropriate algorithm. It then generates outputs directly to the appropriate actuator(s) or the lower level controller.

As for the handling of spatial entities during the reasoning process, the

formal definition of concepts as per the theory is handled by the Tbox of the DL knowledge base. It contains "terminological" space axioms mostly in the form of mereological and topological binary relations (as defined in Section 4.4.2) embedded in the structure of the space ontology. These axioms also provide type definition to spatial objects contained in the Abox which encompasses assertional axioms on the space domain. The rules engine encodes and enforces system-level rules and calculations that affect the domains involved in the CPS behavior. This spatial modeling architecture makes use of rule-based reasoning which encodes rules in the form of *"if...then"* statements. The spatial reasoner (1) checks for (un)satisfiability of propositions constructed with the combination of Tbox and Abox elements in order to ensure consistency of the space knowledge base and, (2) infers new relations between input/existing space concepts and objects.

As for time, tableau algorithms can support the testing and checking of consistency in the database and the construction of a clash-free tree of spatial concepts. Put together, these trees compose triple (RDF) graphs of queryable space concepts. Both Subjects and Objects in triples are convex space regions and Predicates are fully compatible with RCC-8 specification as defined in Section 4.4.2.

4.5 Working with the Java Topology Suite (JTS)

The Java Topology Suite (JTS) is an object-oriented software library providing Euclidian planar linear geometry algorithms in computational geometry. The initial goal of the project that led to the development of JTS was to develop a

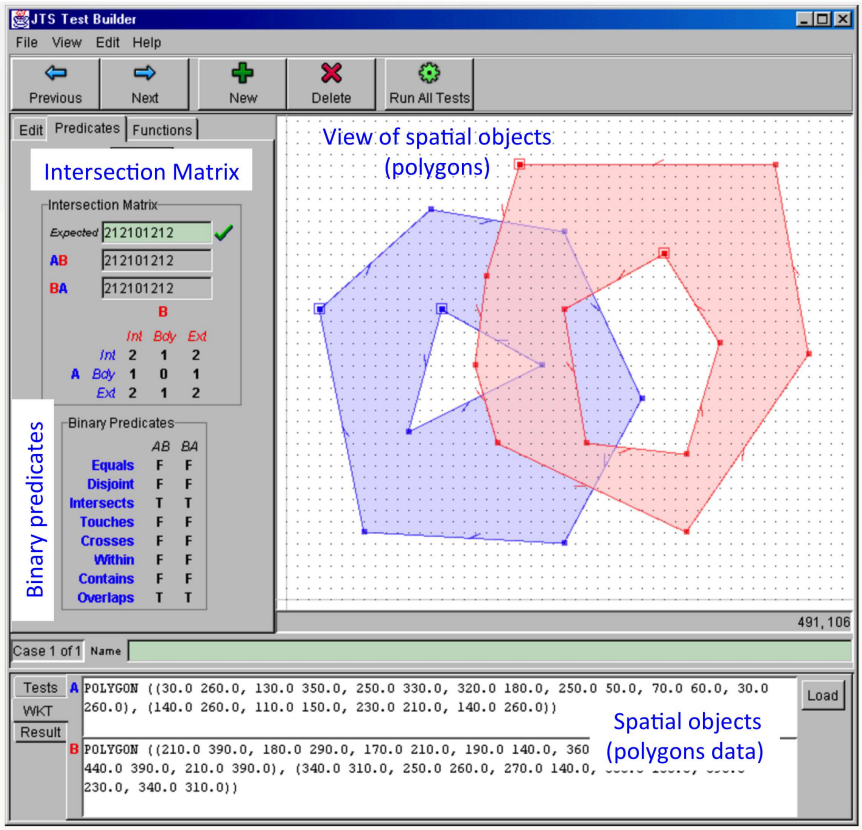


Figure 4.5: An annotated view of Java Topology Suite Test Builder User interface (source: *https* : *//live.osgeo.org/en/overview/jts*$_o$*verview.html*).

Java API implementing the Open Geospatial Consortium (OGC) Simple Features Specification for the Structured Query Language (SQL) [60]. The Simple features specification is an ISO standard (ISO 19125) that specifies mechanisms for storage and access of (mostly) two-dimensional geographical data. The current version of JTS (1.8) - released under the GNU Lesser General Public License - provides a complete, consistent and robust implementation of fundamental 2D spatial algorithms (visit *http://www.vividsolutions.com/jts/JTSHome.htm*).

As a geometry engine, JTS offers several key capabilities including (1) formal definition and representation of all types of geometries such as Point, MultiPoint,

LineString, MultiLineString, Polygon, MultiPolygon, GeometryCollection, (2) geometry methods for establishing spatial predicates, performing overlay operations, and computing metrics such as area() and length() and, (3) geometry processing operations such as line merging, noding & polygonization or simplification. Along the path towards designing and debugging spatial algorithms for JTS, researchers have recognized the central role of spatial visualization. Thus, they have developed *JTS TestBuilder*, an interactive desktop interface enabling users to run tests and experiment with geometry. The intersection matrix of spatial objects is represented using the Dimensionally Extended nine-Intersection Model (DE-9IM) which is a topological model and a standard used to describe the spatial relations of two regions [258]. As shown in Figure 4.5 spatial objects can be visualized and binary predicates (intersection, equals, overlaps, etc.) can also be tested.

The extensive, full-featured, robust, efficient library of spatial operations provided by JTS has made it a cornerstone of leading spatial applications including the Java-based Unified Mapping Platform (JUMP), GeoTools and Moxie Media Internet Mapping Framework. In [61], the author provides a survey of JTS functions and components as well as tips for using JTS as an engine for processing Geometry and its components and APIs for spatial algorithm development.

4.6 Case Study: Spatial Modeling and Reasoning in Action

4.6.1 Case Study Description

Simple Spatial Reasoning. In this example, we consider the system shown in Figure 4.6 a) representing a car (P_0) driving through a work area (S_0). The system is modeled with two spatial entities, a Point P_0 traveling on a linear trajectory Tr_0 that crosses a rectangle S_0. We seek to determine the relative position of the point to the rectangle object as it travels along trajectory Tr_0. From the world prospective and for effective decision making, we want to know whether the following proposition is true or not:

Φ_0 : P_0 is inside S_0.

A usage of the value of this proposition could be to adjust/reduce the speed of the car (P_0) when it reaches the work area (S_0) so it can cross safely.

Formally, this problem can be translated into a variety of questions as introduced in Section 4.4.2 as follows: (a) Mereological : Does P_0 lie within S_0? (b) Logical : Does Φ_0 hold all the time as P_0 travels along Tr_0 ?

The implementation needs simplified geometric models of P_0, Tr_0 and S_0. Their underlining representations as respectively JTS Point, Line and Polygon allow the creation of corresponding semantic representations consistent with encoded spatial knowledge.

Spatial Reasoning with Safety Constraint. System configuration in Figure 4.6

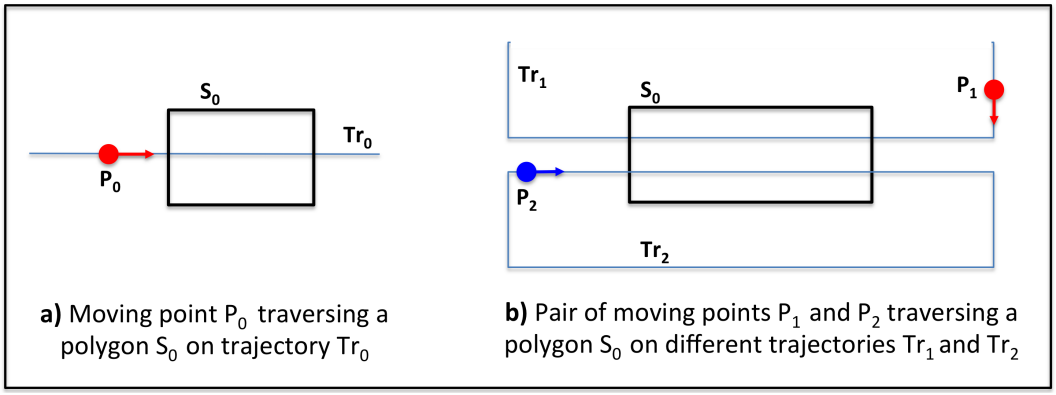


Figure 4.6: Simple spatio-temporal reasoning examples

b) is considered in this case. Now, two cars (P_1 and P_1) compete for the access to the resource (S_1) and only one can use it at a time. The system is modeled using two points P_1 and P_2 traveling respectively on rectangular trajectories Tr_1 and Tr_2 crossing a rectangle S_0. We seek to avoid the violation of the following constraint:

C_{12} : P_1 and P_2 shall never be inside S_0 at the same time.

From the world prospective, we want to know whether the following proposition is true or not:

Φ_{12} : The system is safe under constraint C_{12}.

Formally, this can be translated into a variety of questions as introduced in Section 4.4.2 as follows: (c) Mereological : Do P_1 AND P_2 lie within S_0? (d) Logical : Does Φ_{12} hold all the time as P_1 AND P_2 travel along Tr_1 and Tr_2 respectively?

Keeping the system safe i.e., to satisfy constraint C_{12}, requires not only the control of the dynamic of the P_1 and P_2 but also their coordination in both temporal and spatial domains to prevent the occurrence a the unwanted system configuration.

4.6.2 Spatial Ontology and Rules

Space Ontology. A simplified ontological representation of the spatial domain along with its key rules are needed to solve this case study. The right-hand side of the Figure 4.7 shows an excerpt of the space ontology with a representation of the relationship among classes and properties.

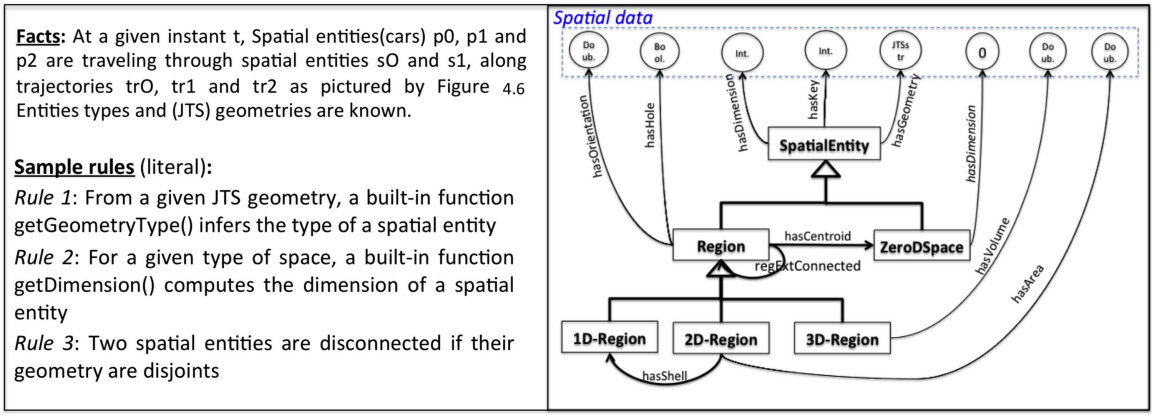


Figure 4.7: Illustration of a simplified ontology of space and sample literal rules

A spatial entity has a few core properties: `hasGeometry`, `hasDimension`, and `hasKey`(unique). `Region` and `ZeroDSpace` are subclasses (specializations) of the class `SpatialEntity`. Similarly, `OneDRegion`, `TwoDRegion` and `ThreeDRegion` are specializations of the class `Region`. The following excerpt illustrates the initial and formal description of a `Region` in the OWL ontology.

```
<owl:Class rdf:about="http://petnga.org/ontologies/space#Region">
    <rdfs:subClassOf rdf:resource="&space-entry;SpatialEntity"/>
    <owl:disjointWith rdf:resource="&space-entry;ZeroDSpace"/>
</owl:Class>
```

A `Region` has an ObjectProperty `hasCentroid` which points to a unique `ZeroDSpace`

as range. A number of DatatypeProperty helps characterize and specify classes. For instance, `ThreeDRegion` can have `hasVolume` as property. Similarly, `hasArea` is well understood to be a property of `TwoDRegion`. A restriction on the range of property `hasDimension` to the value 0 (integer) is necessary to characterize `ZeroDSpace`. We note that, in our ontology, the range of the property `hasGeometry` is a string as shown in the excerpt below. Its value is of the JTS representation of the `SpatialEntity`.

```
<owl:DatatypeProperty rdf:about="http://petnga.org/ontologies/space#hasGeometry">
    <rdfs:domain rdf:resource="http://petnga.org/ontologies/space#SpatialEntity"/>
    <rdfs:range rdf:resource="&xsd;string"/>
</owl:DatatypeProperty>
```

Figure 4.8 (a) shows the detailed description of the ontological class `Region` in the knowledge base along with its defined and inherited subclasses and superclass. The inheritance mechanism between classes is activated at the creation of the ontology via class propagation inferencing. This allows the given class to inherit the properties of the superclass as well. For instance, DatatypeProperty `hasGeometry` and `hasDimension` as well as ObjectypeProperty `inside` are inherited from its superclass `SpatialEntity`.

Facts and Rules for Spatial Inferencing. The left-hand side of Figure 3.7 shows a set of facts and three rules. The facts capture the state of the system conceptualized in Figure 4.6 at a given time `t`. Some of them are synthesized as follows :

- (F0) Entities p_0, p_1 and p_2 are all instances of `ZeroDSpace` class; (F1) Entities

p_0 name is "P_0"; (F2) Entities p_0 initial geometry is "POINT (100 700)" ;

• (F3) Entities tr_0, tr_1 and tr_2 are all instances of `OneDSpace` class; (F4) Entities tr_0 name is "Tr_0"; (F5) Entities tr_0 geometry is "LINESTRING (100 700, 300 700, 700 700, 900 700)" ;

• (F6) Entities s_0 and s_1 are all instances of `TwoDSpace` class; (F7) Entities s_0 name is "S_0"; (F8) Entities s_0 geometry is "POLYGON ((300 600, 300 800, 700 800, 700 600, 300 600)" ;

The selected three rules are as follows:

```
// Rule 1 (R1) : Compute and extract the type of the spatial entity from its geometry ...

[ GeoTypeSpace: (?x rdf:type se:SpatialEntity) (?x se:hasGeometry ?g)
    getGeometryType(?g,?gt) noValue (?x se:hasGeometryType ?gt)
    ->  (?x se:hasGeometryType ?gt) ]

// Rule 2 (R2) : Infer the dimension of the space from its type ...

[ Dim0DSpace: (?x rdf:type se:ZeroDSpace) noValue(?x se:hasDimension ?d)
    -> (?x se:hasDimension "0"^^xsd:int) ]

// Rule 3 (R3): Deduction of "regDisConnected" relationship between spaces ...

[ DC: (?x rdf:type se:Region) (?y rdf:type se:Region) noValue (?x se:regDisConnected ?y)
    (?x se:hasGeometry ?g1) (?y se:hasGeometry ?g2) getRCCRelation(?g1,?g2,?rel)
    equal(?rel,"DC"^^xsd:string) -> (?x se:regDisConnected ?y) ]
```

4.6.3 Spatial Reasoning

The rules are applied to the space ontology to transform the structure of the space semantic graph through inferencing. Similarly as seen for the temporal domain, the ontology is first instantiated with the set of facts (F0)-(F8) and similar ones. Again, Jena is called to extract classes and properties from the loaded ontology, then create semantic-

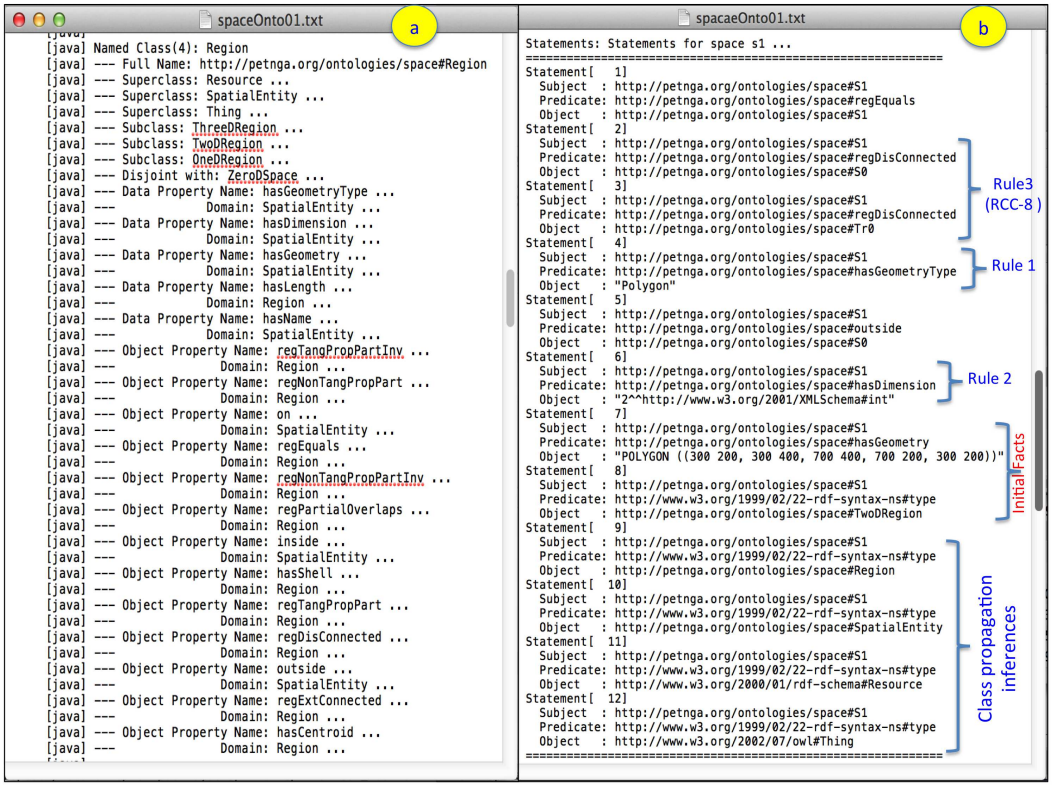


Figure 4.8: A view of (a) the ontological class Region and (b) statements relative to space s_1 in the knowledge base

compliant spatial data and finally add the facts as statements in model of the semantic graph for the space domain. Facts are encoded as statements the same way as the ones introduced in Section 2.6. For instance, the following statement is a translation of fact (F2).

```
Literal geomP0 = modelSpace.createTypedLiteral("POINT (100 700)", XSDDatatype.XSDstring );
Statement pOGeometry = modelSpace.createStatement( pO, hasGeometry, geomP0 );
modelSpace.add ( pOGeometry );
```

The excerpt in Figure 4.8 (b) shows all the statements in the knowledge base (which contains the semantic graph) for the spatial entity s_1 after the spatial rules are added and executed. The result of Rule 1 is shown in `Statement 4` with the reasoner accurately

inferring `Polygon` as the type of the geometry of this entity. Similarly, Rule 2 infers its dimension to be 2. Rule 3 relies on RCC-8 to establish that this region is disconnected (`regDisconnected`) from the other two regions in the knowledge base at the time i.e. $\texttt{Tr}_0$ and $\texttt{S}_0$. Given the open world assumption of ontologies, the remaining relationships do not appear when there is no match in the knowledge base. We also note, in `Statement 11` - `Statement 13`, the appearance of new types for entity s_1 from the initial one stated by fact (F6). They result from the class propagation rules built in the ontology and enforced by the Jena API at its creation.

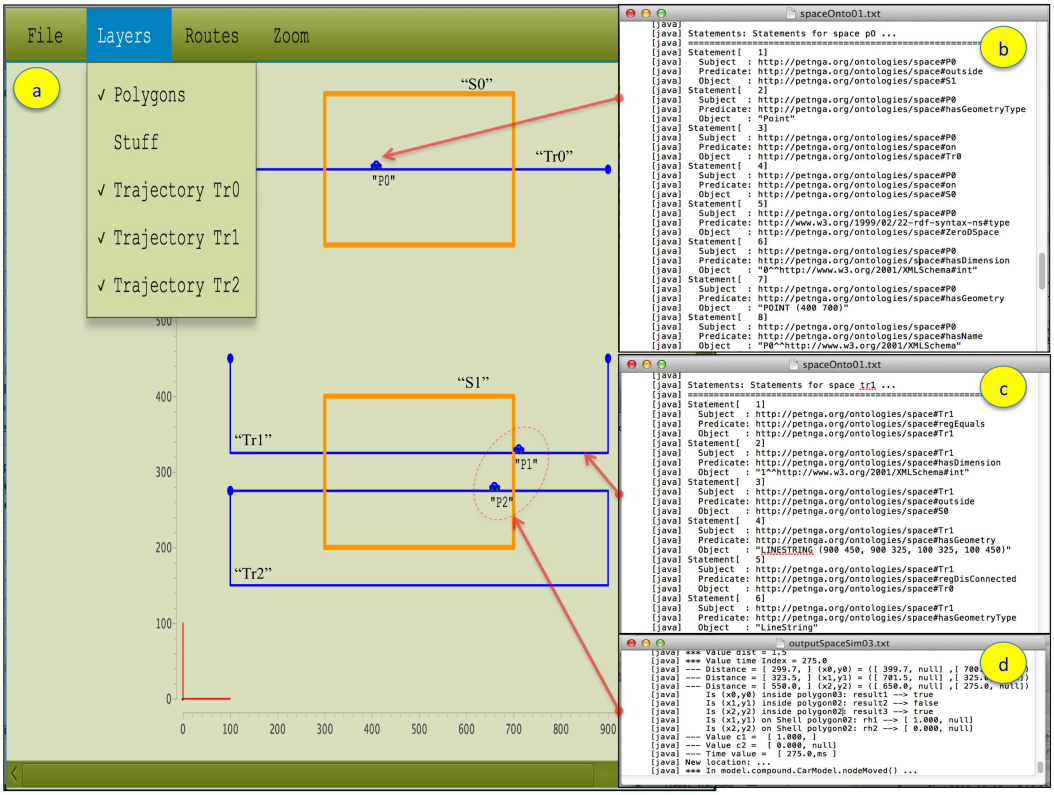


Figure 4.9: Physical and semantic model views of simulation of safety constraints

Figure 4.9 (a) shows the physical model of the system conceptualized in Figure 4.6. It's realized thanks to the capabilities of Whistle (more in Chapter 6). The top right hand side of the figure shows the corresponding semantic description of the situation in

the knowledge base.

In excerpt (b), we note in `Statement 7` the change in the value of the geometry of entity $\mathtt{p}_0$ from the initial fact (F2). This is consistent with the notion that the entity is dynamic and moving along $\mathtt{tr}_0$ thus its position (which is also its geometry in this case) will change as the entity moves. Also, it appears in `Statement 4` that proposition Φ_0 in Section 4.6.1 returns the value `true`.

In excerpt (c) as opposed to $\mathtt{p}_0$, the geometry of $\mathtt{tr}_0$ hasn't change as the entity is fixed (see `Statement 4`). As a `Region`, the RCC-8 predicated are evaluated and the result appears in `Statement 1` and `Statement 5`.

Excerpt (d) illustrates the tracking of the actual location (sensor measurements) of the entities in the physical world and its usage for control and enforcement of constraint $\mathtt{C}_{12}$ introduced in Section 4.6.1. The circled area containing both cars shows $\mathtt{p}_1$ waiting at the border of $\mathtt{s}_1$ for $\mathtt{p}_2$ to leave before it crosses the conflict area too. This translates into proposition Φ_{12} returning the value `true` as well.

Chapter 5: Framework for Ontological Modeling and Decision Support

5.1 Introduction

This chapter builds upon the semantic foundation and MBSE procedures described in Chapters 1 through 4 to take a first step toward mitigating present-day deficiencies in CPS analysis and design. We lay down the foundational building blocks to support the development of determinate CPS models, with strong temporal and domain-specific semantics strengthening model-driven approaches to CPS design. Our focus will be on the data and information processing layer of CPS modeling, with a particular attention to procedures and mechanisms for producing determinate, provable and executable CPS models. We introduce and describe an innovative ontological framework, and illustrate the structure and phases of construction of a knowledge modeling and decision support framework for CPS (CPS-KMoDS). The framework offers some flexibility in its implementation, for example, for the selection of tools and type of tasks targeted by the model. System dependability characteristics, especially safety, are viewed as multi-domain models that drive the evaluation of decision tasks and, as such, the development of the ontological framework.

Section 5.2 presents strategies to address semantics challenges identified in Chap-

ter 1 as well as key requirements to be satisfied by CPS-KMoDS models. DL extensions to the mathematical foundations identified in Section 2.3 play a central role in reasoning tasks. In Section 5.3, the proposed framework is introduced and its construction process described. A prototype implementation based on Semantic Web technologies and Jena Rules and Reasoning is presented in Section 5.4. We exercise the framework through the development of a reasoning system to support decision making for autonomous cars passing through a traffic intersection controlled by smart traffic lights.

5.2 CPS Knowledge Modeling and Ontologies

5.2.1 Requirements on CPS Models for Decision Making

A well-designed model contains just enough detail to answer the relevant questions and nothing more. For the purposes of this work, the main task at hand is support for decision making, which in turn, drives the need for the development of models that are determinate, provable and executable. The details are as follows.

1. **Determinate:** A model is determinate if it provides answers to questions that are certain and conclusive. For the design of CPS, it is well known that physical processes are not determinate. Similarly, on the cyber side of development, the use of threads as a dominant sequential model of computation to concurrency results in models that are non-determinate [153]. The long-term challenge is to counter these realities by "dynamically changing" programming models so that their correct execution always produces acceptable behaviors at subsystem I/O [158]. This capability will ease the modeling, simulation and verification of non-functional requirements and dependability properties with safety as one of the most important.

Given the restrictions and intrinsic weaknesses of computer systems, this is not an easy task [149]. However, we ought to be able to start by producing well-defined, determinate models and progressively move toward stochastic ones along with ways to deal with uncertainties.

2. **Provable:** A model is said to be provable if it has the capability to establish the validity (or truth) of assumptions. For the design of CPS, the development of provable models is complicated by the heterogeneity of physics, domains, and abstractions emanating from different types of models. Still, with safety at the heart of system characteristics, the precise meaning of models is required; thus, the need for formal semantics and formal descriptions of models that keep unambiguity away.

3. **Executable:** For our purposes, a model is said to be executable if it is formal enough to be processed by a machine. Complicating factors include data and information emanating from multiple distinctive sources, and the need for evaluation of system behaviors that are dependent on multiple physics and multiple abstractions. See Figure 1.8. From this perspective, the CPS model will be similar to a computer program that provides a precise and concise description on how data can be cast into a representation to support decision making. For this process to work well, the underlying modeling language should be decidable in the sense that the designer should be able to automatically determine model correctness and the point of program termination. Unfortunately, standard languages such as Fortran, C, C++ or Java are not decidable, as demonstrated by the unsolvability of the Halting problem [242, 270]. Therefore, to move forward, some restrictions are needed to achieve decidability of a problem formulation casts in one of these languages.

4. **Support for physical quantities:** We observe that if the cyber has an improved ability to understand what is happening in the physical world, then the quality of decision making in the CPS will be improved. Thus, an important capability has to be the design of cyber that can reason with physical quantities, dimensions (e.g., mass, length, time, voltage) and units, time and space. There is a need for CPS models to capture and handle the representation, conversion and computation of physical quantities. CPS models should be able to represent and distinguish (during processing) both dimensions and units. For instance model supportive semantics should clearly establish distinction between length and mass as dimensions (1 meter and 1 kilogram) and units within the same dimension (e.g., 1 minute and 1 second) while properly handling units conversions.

The modeling ecosystem should provide the appropriate structure and constructs to effectively and efficiently deal with these requirements. Together, these features will provide the foundation to formally prove (or not) the satisfaction of systems safety and non-functional requirements, the ultimate target being to obtain correct-by-construction designs. However, despite the numerous strengths of existing approaches, resulting models lack several of the properties and characteristics needed to satisfy the requirements identified here. Major weaknesses include the lack of support for the physical aspects of CPS and ontological modeling of time, which is one of its critical metadomains. The CPS-KMoDS framework introduced in this chapter aims to mitigate these gaps with the use of models and tools based upon ontological and logic-based mathematical foundations introduced in Chapter 2.

5.2.2 Tackling Semantic and Safety Challenges in CPS Modeling and Analysis

Addressing Semantic Challenges. Some researchers have investigated ways to address these challenges with mixed success. In Derler [66], a landscape of technologies ranging from hybrid systems modeling and simulation to concurrent and heterogeneous models of computation (MoC) is presented. The use of MoC in Ptolemy II [41] is possible thanks to well-defined semantics for concurrency and communication between actor-oriented component models. However, despite its many computational advantages, the use of superdense time models [45,177] for timing is not intuitive for system modeling. Jensen [135] builds on these foundations to propose a step-by-step methodology for model-based design of CPS. This contribution addresses challenges in development of the aforementioned abstraction layers, but there is no explicit mention on how to handle non-functional requirements in the broken chain of models produced by the design process. Bhave [36] proposes an architectural-based approach centered on an "architectural view" that encapsulates structural and semantic correspondences between the model elements and system entities represented at multiple layers of abstraction (physical, control, software, hardware). While the mapping between various views enhances reliance of the run-time base architecture, the underlying process remains manual and error prone, especially as the size and complexity of a system grows.

Strategy for Addressing Safety Challenges. It is evident from Figure 1.8 that system safety properties are critical at all abstraction layers, which makes it a permanent concern for any CPS designer. Due to the presence of physical-related elements and concepts in

the physical and platform abstraction layers, both are obvious subjects of safety concerns. One way that safety concerns can become an issue in the software abstraction layer is through deadlocks, which in turn can lead to unsafe system configurations. The rationale here is that timing (from the physical world) in models at this abstraction layer is not a simple performance or quality factor for the software but a design correctness criterion. As a result, the determination as to whether or not the system operates safely at any point in time requires consideration of all of the relevant aspects across the participating domains, physics, and abstraction layers. Fortunately, all CPS share some commonality in the ways they process information [178]. This is illustrated in Figure 1.8 as the so-called "commonality of information" that crosses all domains and abstraction layers of the system design. The basic idea is to design a data structure that encapsulates the relevant knowledge of the CPS of interest while providing the foundation for meaningful construction of models. We would like to provide a mean to structure, organize and formalize that knowledge, and address the challenge of modeling aspects of the system response related to the evaluation of non-functional and safety requirements. The premise here is that these safety properties and non-functional requirements can be formulated as decision problems with true/false or yes/no solutions.

5.3 Framework for Modeling CPS Knowledge and Reasoning Support

5.3.1 High Level Architecture

Closing the knowledge gap in MBSE for CPS requires a modeling and design backbone infrastructure that provides the following capabilities.

1. Mathematical foundations for across domains integration,

2. Formal procedures for handing meta domains critical to system "ities (e.g., reliability, safety) and,

3. Co-design of control algorithm and embedded platform for system smartness.

As shown in Figure 5.1, our research aims at providing those capabilities. The CPS world – in which lives the CPS of interest (target) – is abstracted by models that both represent the target and the theories governing its behavior. This double function of CPS models enables them to capture both the phenomena and data [86]. This capability increases the likelihood of uncovering and observing emergent behaviors during design.

Domain theories support the development of domain ontologies and meta-modeling languages. Here, we consider formal modeling languages with strong semantics that do not provide room for ambiguities. System's operational context and expected level of accuracy and precision dictate both the level of detail of domain ontologies and the right granularity for the meta-domains. Also, they influence the selection of the domain theory for the problem at hand. However, when it comes to CPS, not all domain theories and calculus are created equal. Some of them, especially temporal and spatial ones, could lead to undecidability.

Equipped with these powerful mathematical foundations, models are provided with means to interpret laws and axioms of relevant domain theories and their combination in the context of a given CPS design problem. We move requirements from its natural problem space to the solution space by translating then into formal specifications supported by corresponding requirements model (mapped to its semantics). Thus, the resulting design flow is streamed and the co-design of both the physical and cyber is rendered possible. The interaction between the control algorithms and the embedded

platform is manageable through a set of interface variables [112]. The composition of the system components/subsystems and appropriate tracking, collection and gathering of data will enable the observation and analysis of system level properties such as safety or performance. Moreover, the interfacing of our framework with an optimization platform could enable design space exploration at low cost.

5.3.2 Overview of the Framework

The global context for the design and construction of this framework is knowledge-enabled models for complex heterogeneous systems, such as CPS. The pathway for this task involves many steps including automated data acquisition, transformation of data to knowledge, and finally the creation of models that are reusable, provable and executable. The first potential use of these models is for system behavior and safety analysis. They can also act as middleware for CPS systems. To achieve these purposes, the CPS-KMoDS framework relies on the composition of domain-specific ontologies (DSO) along with corresponding knowledge bases (DSKB) on one hand and, domain-specific semantics extensions, an integrator and the CPS application on the other hand. The components of the framework are organized into layers as shown on Figure 5.2. Thus, we describe in the next section the different layers of this architecture, and how the elements interact together to produce decidable CPS models with regard to the requirements identified in Section 5.2.1.

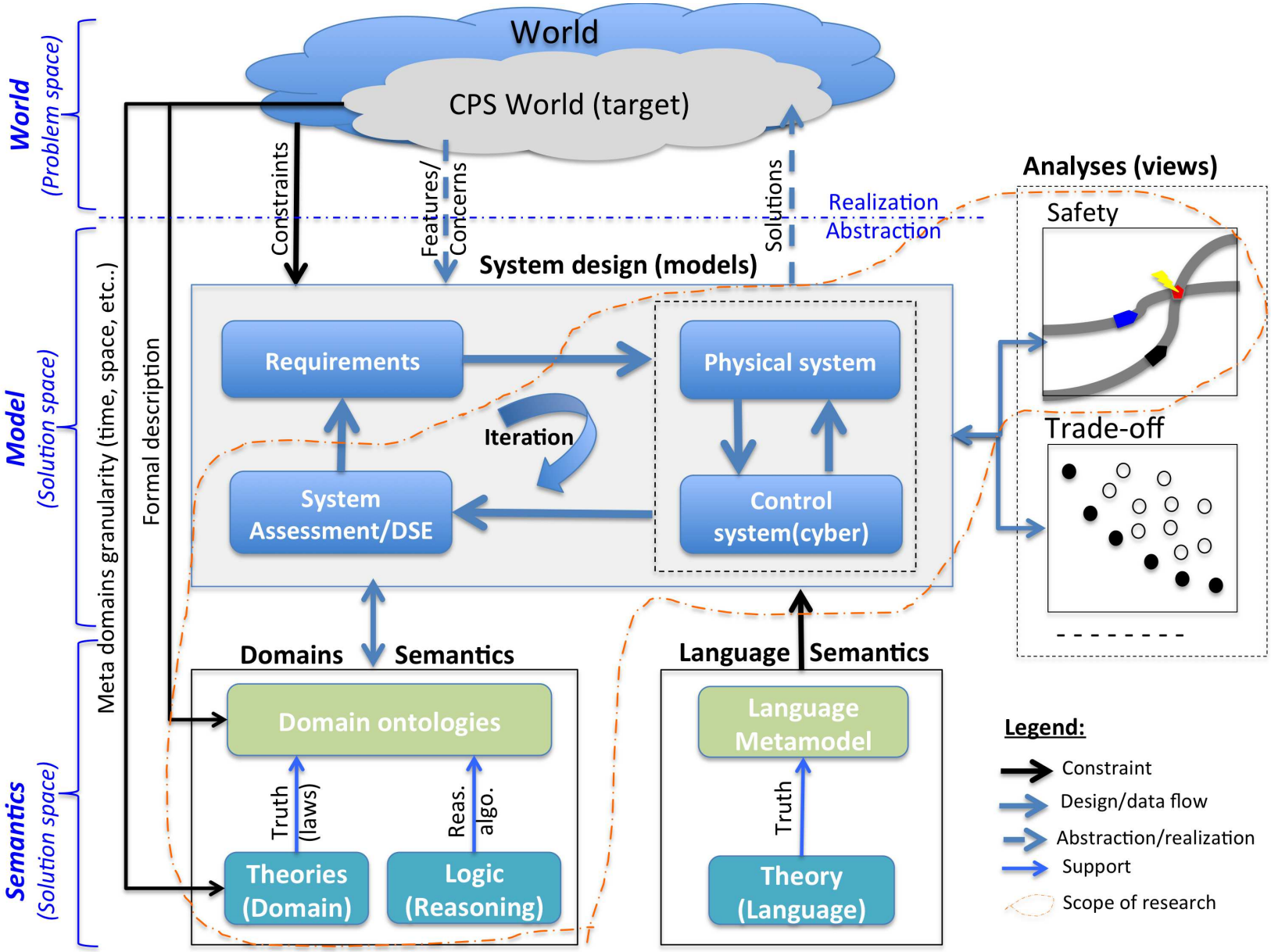


Figure 5.1: High level architecture of a framework for semantic-driven model-based development process for CPS

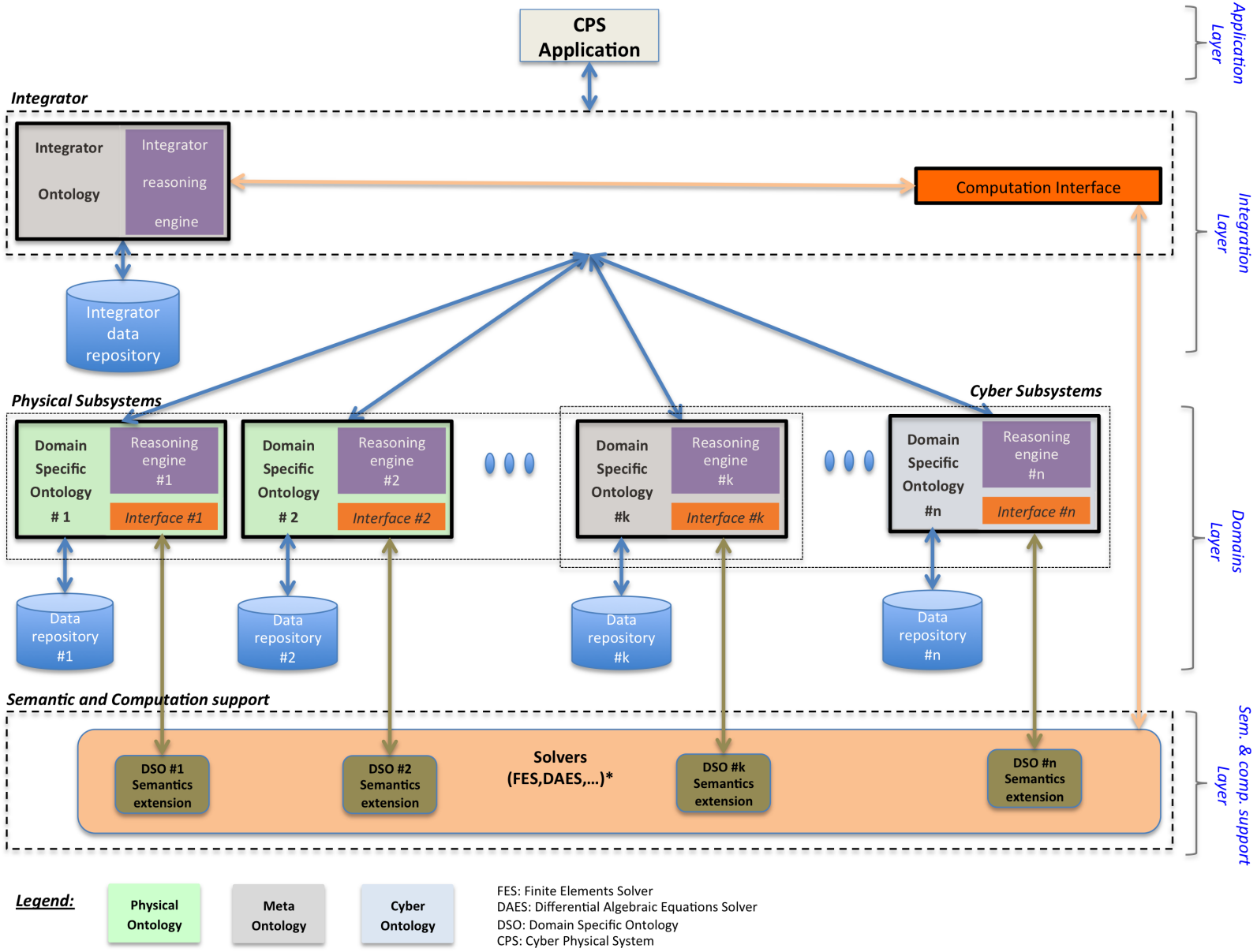


Figure 5.2: Architecture of the CPS-KMoDS

5.3.3 From Data to Knowledge: Domain-Specific Ontologies & Semantics

The domains layer is a modular piece at the center of the CPS-KMoDS architecture. It covers the participating domains and disciplines (see the columns of Figure 1.8), thus, represents concepts relevant to the CPS under study. To completely capture and represent the domain knowledge for CPS, we go beyond simple standalone ontologies toward an architectural structure spanning the bottom two layers of Figure 5.2. The elements of these layers are (1) domain-specific ontologies, (2) data repositories and, (3) semantic extensions and computation support. The details are as follows.

Domain-Specific Ontologies. Each domain is formally defined and described by a light, modular, and reusable basic ontology that captures its core concepts and properties. Then, it is extended with application-oriented concepts and properties. This approach to ontology specification is consistent with the TBox definitorial as introduced in Section 2.3.2 and formally defined in Appendix A. In the absence of instances these ontologies are reusable across applications. Laws and constraints of the domains are captured and translated as rules in domain-related *rules engines*. In order to provide support for complex computations and also to enforce semantics of a given domain, an *interface* to the relevant computational platform is needed by the reasoner.

Our framework employs three types of domain-specific ontologies.

1. *Physical ontologies.* These are ontologies of physical subsystems involved in the CPS of interest, for example, an automobile, a building or an aircraft.

2. *Cyber ontologies.* These ontologies describe the cyber part of the CPS are under this

category. A software is an example of such domain.

3. *Meta ontologies.* Meta concepts such as time, space, or privacy that are relevant to the system are captured and described by this category. Because of their cross-cutting nature, they can apply to either the physical or cyber worlds, or both.

Data Repository: The data repository contains instances of the concepts defined in the ontologies. As the assertion component of the architecture, instances are interpreted as the ABox in the DL formalism. It is important to note that this interpretation operates under an "open world assumption" as opposed to a "close world assumption" of databases. Thus, the reasoner is prevented from drawing erroneous and invalid conclusions from the facts in the knowledge base. Control mechanisms embedded in the rules engine ensure that any data available in the repository is correct. These measures are particularly important, as CPS are safety critical systems and the decision made has to be the right one (always) in order to guarantee system safety.

Semantic Extensions and Computational Support: As shown in Section 2.3.1, the $\mathcal{SROIQ}$ DL is equipped with appropriate formalisms to handle concrete domains – these are pre-defined interpretation domains for which semantics of datatypes are invariant (i.e., the same no mater the interpretation). Therefore, the development of this framework with an ontology language backed by this DL or equivalent will provide similar support.

Unfortunately, supported concrete sets such as real, integer or boolean that are computation friendly can miss essential information (e.g., dimensions and units) from the physics of the domain of interest. Support for reasoning with physical quantities can be made as needed, and made available to the reasoner through its interface. Hence, the corresponding computational platform will be able to process physical quantities-based

datatypes. These orientations put our framework at the forefront of the efforts for a more "physicalization of the cyber world" in the sense of Lee [157].

5.3.4 From Knowledge to Model: System Integration

The pathway from knowledge to models is defined by a systematic build-up of knowledge models from domain-specific ontologies. Ontologies from disparate domains need to be merged and integrated with ontologies that represent concepts from cross-cutting concerns, such as time.

Assembling Ontologies. Domain-specific ontologies along with rules engine and semantic support are good foundations to domain-oriented system modeling, as described in the previous section. In the context of CPS modeling, however, stand alone formalizations of a sub-domain do nothing more than provide a formal description of the domain and means for the designer to test proper low-level interactions between the different modules at the domain level. Even though the latter step is very important, it is not enough. There is a need to reuse the various domain ontologies in a coherent and correct assembly. This has to be done in a way that properly renders the CPS of interest while preserving the decidability of the underlining DL formalism. Several researchers [48, 252] indicate that the following techniques provide a pathway for moving forward.

1. *Merging.* Ontologies for similar domains are merged into one single coherent ontology.

2. *Alignment.* Complementary domains ontologies are linked, resulting in two or more ontologies.

3. *Integration.* Ontologies from different domains are merged in one single ontology.

The categorization of DSO (as shown in Section 5.3.3) prevents the designer from introducing overlaps between ontologies during their development. This is not always guaranteed as concepts and properties can be repeated in different ontologies. Still, the CPS model has to be viewed as a unified domain, thus the need for a single ontology backing the model. This leaves us with "ontology integration" as the appropriate pathway for assembling individual ontologies in the CPS-KMoDS framework. The CPS ontology is created by merging all ontologies (including the integrator) under a single umbrella ontology that is checked for consistency before any further use.

Integrator Ontology and Extensions. The integrator is created to capture, represent and translate CPS properties and concepts that are not part of a specific subdomain. Concepts in this ontology are mostly from the individual DSO. The integrator has its own rules engine that translates the constraints and laws applicable to the CPS of interest. It also handles system metrics and control parameters, including decision rules capable of determining system safety state at any point in time. There is no need for semantic extension to support this ontology as it's not related to a specific concrete domain. Depending on the problem, one might need the system rules engine to interface with external solvers, for example, to handle complex calculations such as differential algebraic equations (DAE) or finite element analysis (FEA). A computation interface augmented with proper semantic translation capabilities is charged with linking both modules. However, the effectiveness of such computation platforms are dependent on the performance of the implementation hardware. This could significantly affect on-going decision tree exploration in the rules engine in a context where timing is a design correctness criterion, as seen in Section 5.2.2. For those cases where the granularity of modeled time is not appropriate for the selected computation support platform, solvers can be replaced by set

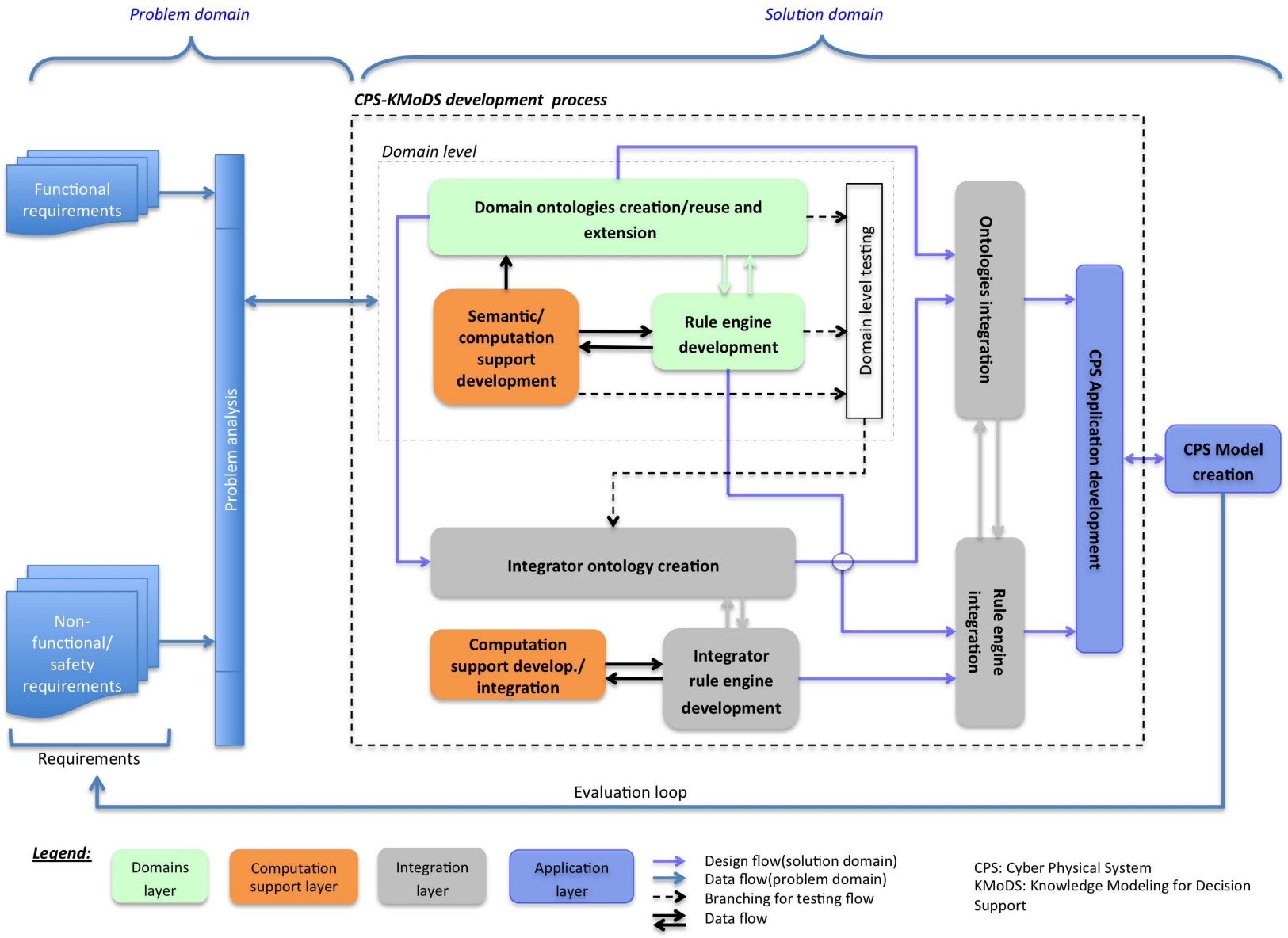


Figure 5.3: Proposed flow chart for development of the CPS-KMoDS framework.

of lookup tables. These lookup tables will encode, with a high level of precision, solutions to the system's equations.

5.3.5 Reasoning for Decision Support

Reasoning is concerned with the use of inferencing techniques to draw conclusions from a set of premises. In the proposed framework, decision trees are translated into sets of logical rules that can be evaluated through the use of reasoning strategies.

Choosing a Reasoning Approach. Generally speaking, reasoning techniques can be of three types which are logical [57, 180], heuristic [55, 220] and, ethical [12, 70]. Because of their weak underlying formalisms thus, high risk of undecidability, the last two approaches are weak and not good candidates for our framework. Therefore, moving forward, we only consider automated mathematical logic-based reasoning approaches, with a bias towards those enabling automated theorem proving. We would like every logical inference to be checked all the way back to the fundamental mathematical axioms in order to ensure model provability. Thus, some critical capabilities are needed to the rule-based reasoning approaches that we adopt for our CPS-KMoDS framework.

Rule-based reasoning for CPS-KMoDS. Rule-based reasoning is the classical approach to logic-based reasoning, where the knowledge-based system is developed by deduction, induction, abduction or choices from a starting set of data and rules. The main components of a rule-based reasoning system are: (1) the rule base, (2) the inference engine, and (3) a variety of miscellaneous integration components.

The rule base corresponds to a set of rules applicable to the (sub)system of interest. Rules are of the form of "*If... then...*" statements. Each rule is made of a *body*

that contains all premises or conditions and a head that states the conclusion(s) when the conditions are satisfied. In CPS-KMoDS, rules are written and used in the following three ways.

1. *Forward chaining (materialization).* The rule base is scanned and heads are pre-computed and stored. Conditions are evaluated one at a time, from left to right. The evaluation stops any time a condition is not satisfied and the rule is not fired. The CPS-KMoDS Integrator rules engine uses this method to hook to and access external lookup tables when needed.

2. *Backward chaining (query-rewriting).* The computation of the head of the rule is done on-demand with a minimal index storing. When the head of the rule is not called in an instance of an execution chain of the rule base, the given rule is not evaluated. This approach appears to be the most indicated for writing rules in our framework, especially when it's used as middleware.

3. *Hybrid chaining.* This approach combines the previous ones in complex rules designed to take full advantage of both methods.

Inference engine provides mechanisms supporting the use of the ontology language and allowing for additional facts to be inferred from available data in the repository and class definitions. Within the context of CPS-KMoDS, semantic reasoners are the actual concrete code objects that perform the inferencing task. The chosen ontology language for our framework is OWL, which is $\mathcal{SROIQ}$-backed and decidable, as shown in Section 2.3.1 and Appendix C. For practical applications of reasoning, any of the leading OWL DL reasoning tools introduced in Section 2.4.4 such as Pellet or the Renamed ABox and

Concept Expression Reasoner(RACER) can be used. They can be plugged into actual implementations of the CPS-KMoDS framework.

Miscellaneous components for the reasoning system include a temporary working memory to store information that is in-transit between different computation cycles, and connections to other parts of the framework. The latter are essentially internal links to TBox, ABox and possibly external connections to semantic extensions and computation support systems through interfaces(when needed), as shown in Figure 5.2.

In our framework, reasoning engines are implemented and tested for each of the DSOs involved in the CPS of interest. They are integrated into a system rules engine along with the integrator rules engine. This operation mimics the above-mentioned ontologies integration process. Figure 5.3 employs the architectural component shown in Figure 5.2 and synthetises the development process for modeling and analysis of CPS behavior and properties in the CPS-KMoDS framework.

5.3.6 Dimensional Reduction for Decision Making in CPS

Dimensional Analysis Foundations. Dimensional analysis(DA) is the study of the relationship between physical quantities through the identification, comparison and tracking of their fundamental dimensions (e.g., mass, length, time, voltage) and units (e.g., grams, pounds, miles, meters) during calculations and transformations. DA originates for the need identified by earlier physicians to make physical laws independent of the units used in the measurement of the physical variables involved in their formulation. This has led to the conclusion that those laws should be formulated in term of homogeneous equations consistent with their multiple (possible) units of measurements, a result that has been

formalized through the Buckingham Π theorem.

Theorem 1 (Buckinghan's **Π**). *When a complete relationship between dimensional physical quantities is expressed in dimensionless form, the number of independent quantities that appear in it is reduced from the original n to m = n - r, where r is the maximum number of the original n that are dimensionally independent. It's also the rank of the dimensional matrix. The rows of this matrix are made of all variables of the system while columns contain all independent dimensions present in the system.*

The proof can be found in [38]. Theorem 1 guarantees that every physically meaningful equation involving n variables can be equivalently rewritten as an equation of n - r dimensionless parameters. It also provides an approach for computing the dimensionless parameters from the physical variables following a simplified but sound procedure.

Simplified Dimensional Analysis. The DA process consists of building a similarity transforms S and its inverse S^{-1} between the dimension space $\mathcal{X}$ formed by the physical variables involved in the problem i.e. $(x_1, ..., x_n)$ and the dimensionless space Π made of dimensionless variables $(\pi_1, ..., \pi_m)$ as shown in Figure 5.4(a). The formulations in dimension and dimensionless spaces are said to be "physically similar" in the sense of Buckingham in [42]. Figure 5.4(b) summarizes the dimensional analysis procedure which can be broken as follows.

Dimensional equations and dimension matrix. Dimensions reduction or elimination is done through nondimensionalization, a process which involves scaling physical quantities by characteristic units of the system object of the study or natural units. Formally, physical quantities are expressed as product of the basic physical dimensions and corresponding base dimensions in SI standard-compliant symbols i.e., length(L), mass(M),

time(T), electric charge(Q), and absolute temperature(Θ), each raised to a rational power according to the given physical equations. The dimensions formed from a given collection of the basic SI standard-compliant symbols, form an *abelian group* i.e. one for which the order of application of the group operation to two group elements does not affect the result (axiom of commutativity). These relations form the dimensional equations from which the dimension matrix (D_{nr}) is extracted. The latter is an (n x r) matrix which content is made of the rational power of individual physical quantity symbol as formulated in the dimensional equations.

Partition and Dimensionless matrix. The dimension matrix is organized and partitioned into two block matrices: B_{mr} for the dependent variables and A_{rr} for the independent variables. In the definition of these matrices, we have :

- n = number of physical variables,
- r = number of dependent variables (= number of base dimensions) and,
- $m = n - r$ = number of independent variables (= number of π groups).

The completeness of the list of physical variables is checked by making sure $|A_{rr}| \neq 0$. This will also guarantee that A_{rr} is invertible. The problem can now be cast as expressing each of the m dimensionless variables or π-groups as a function of an individual independent variable and appropriate dependent variables. The matrix product $-(A_{rr}^T)^{-1}B_{mr}^T$ provides the solution matrix to this problem as part of the general solution ϵ as follows.

$$\epsilon = \begin{pmatrix} I_{mm} \\ -(A_{rr}^T)^{-1}B_{mr}^T \end{pmatrix} \tag{5.1}$$

ϵ is a matrix of size (n x m), with I_{mm} being the identity matrix.

Advantages and Limitations of DA There are multiple advantages in the use of DA

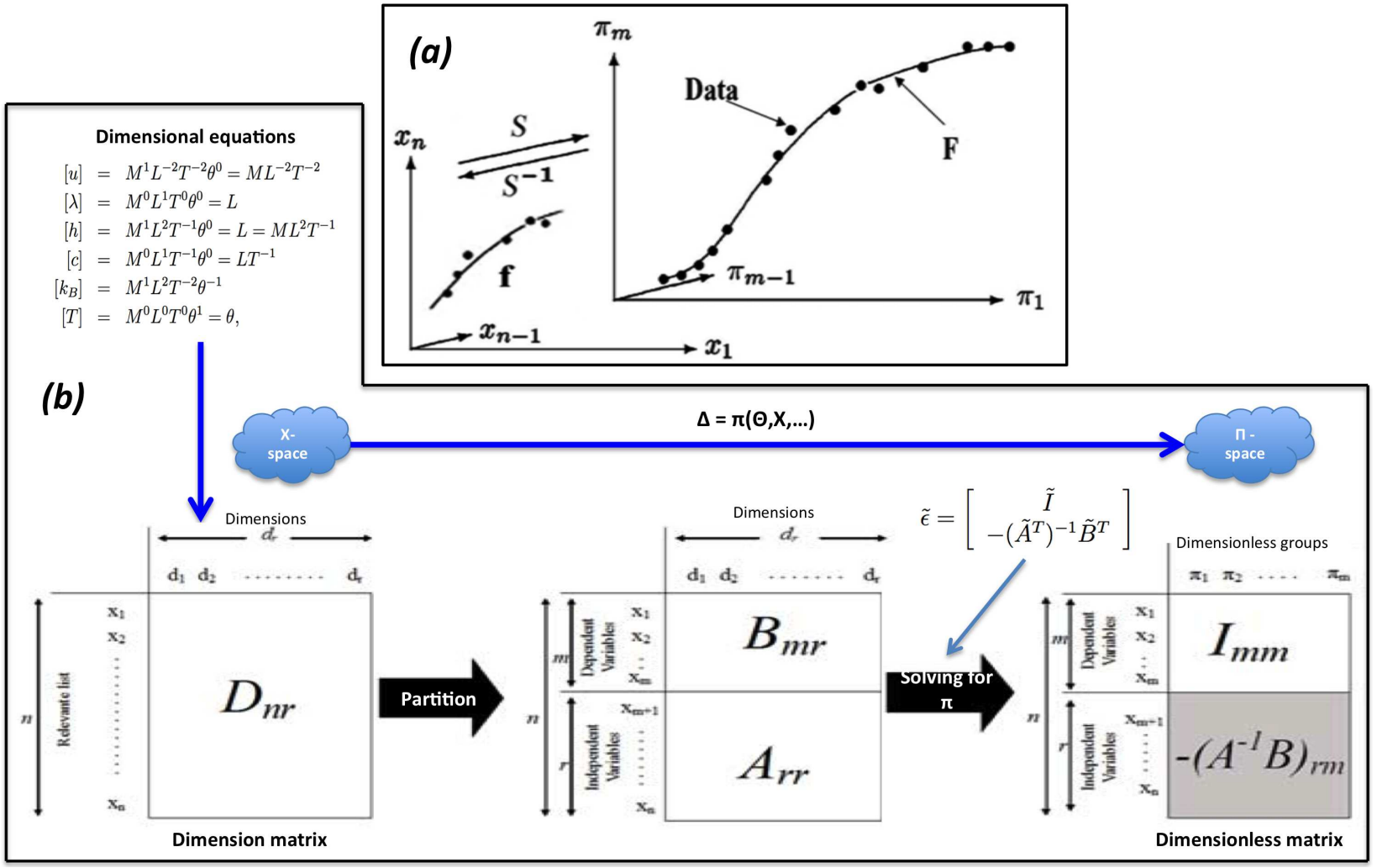


Figure 5.4: Summary illustrations of dimensional analysis procedures. (a) Similarity transforms functions S and S^1 between dimension space $\mathcal{X}$ and dimensionless Π (adapted from [230]) (b) Construction of the dimensionless matrix from the dimensional one using rank-preserving operations (adapted from [264]).

procedures and techniques in system analysis and design.

Reduced Number of Variables and Workload. Example applications mostly in physics and chemistry demonstrate the effectiveness of DA in reducing the number of problem variables from n (in the physical space) to $m = n - r$ (in the dimensionless space) [42, 264]. The reduction of the number of variables that must be specified to describe the behavior of the system or a phenomenon at hand leads to a huge simplification of both the problem and solution spaces. As pointed out by Hanche-Olsen [113], the complexity of theoretical analysis and experimental design are significantly lowered.

Similarity Laws. DA provides mechanisms to uncover and formulate similarity law for the physical phenomenon object of the study or analysis. Also, under certain conditions, it enables the establishment of equivalence between physical phenomena that are different [253].

Out-of-scale Modeling. DA enables performance analysis and study of larger scale systems for which a "geometrically similar" smaller model can be built and experimentally tested. Full-scale performance is established from measurements at the small scale using relationships between the two systems established by similarity laws above.

Remark 5.1. *(Pitfalls and Limitations of DA).* For the DA to yield all the above-mentioned benefits, the analyst should be careful to avoid some of its pitfalls inherent to the decisions and choices made during the process [253]. One should make sure that the selected set of independent variables is complete in order to avoid erroneous DA results. Similarly, superfluous independent variables complicate the results unnecessarily. One of the most important limitations of DA procedures, especially ones based on factor-label methods are their inability to properly handle dimensions where relationships between

physically equivalent units is defined by "affine transforms" (i.e. of the form $x \rightarrow ax + b$) rather than "linear transforms" (i.e. of the form $x \rightarrow ax$) that are common between physical units. One such example, for the temperature dimension, is the conversion between degree Fahrenheit (°F) and degree (°C) Celcius: $\theta[^\circ F] = (9/5)\theta[^\circ C] + 32$.

When Dimensional Analysis Adds Value. Built-in support for dimensional analysis is valuable when the problem analysis step uncovers multiple and complex heterogeneous dimensions from the various physics involved in the CPS of interest. It's also helpful in taking advantage of any opportunity to formulate and solve the system measures of effectiveness using dimensionless metrics. In traditional dimensional analysis, each dimensional system comprises a number of base dimensions that are sufficient to define the magnitude - also called unit - of any numerically expressible quantity. In order to address these challenges, we depart from standard systems engineering design processes by augmenting the problem domain with the task of performing dimensional analysis, which lays down the foundation for optimizing the reasoning process downstream. The application of the proper dimensional analysis procedure to the initial decision tree of the problem actually reduces the complexity of the decision problem by transforming the amount of independent variables in the physical space $\mathcal{X}$ into a reduced number of dimensionless variables in the dimensionless space Π. The systematic reduction of the number of dimensions in the problem to a very few base dimensions is an important result of this step. Our approach will customize the transformation to fit the needs and specificity of the problem and the CPS at hand.

Multi-Level Decision Trees. The dimensionless variables resulted from the dimension analysis/reduction will be used to create a generalized system level decision tree for the

problem. Specifically, the decision problem characterizing system safety can be formulated as a tree that is a literal translation of system's sequence of behavior/actions/states that lead to the realization or demonstration of the satisfaction (or not) of its safety. Thus, in this framework, CPS safety and non-functional requirements are cast as a multi-level decision tree, and then translated into the system rule base as an ordered set of rules. We define the size of a decision tree as the total number of possible outcomes. Thus, the size of a decision tree with n decision options is 2^n. Any new node will multiply its size by 2 (from 2^n to $2^{n+1} = 2 * 2^n$), thereby increasing the complexity of the decision tree. The complexity of the decision problem and the participating datatypes is also increased with the presence of physical quantities. This occurs because datatype instances of physical DSOs carry the chosen dimensional system for representing the underlying physics. Each dimensional system comprises a number of base dimensions that are sufficient to define the magnitude (also called unit) of any numerically expressible quantity.

This approach supports the assessment of safety as a system level property. Also, reasoning tasks are simplified as replacement of dimension variables by dimensionless ones alleviates computational complexity by taking away the need for handling units in every single operation. Moreover, this opens the door to the analysis and visualization, at various levels of abstractions, of system properties as shown in the right-hand side of Figure 5.1. Navigating the dimensionless decision tree, tracking and extracting the compounded variables will enable this capability.

Dimensional Analysis for Reasoning Optimization. Under the assumption of the use of rule-based reasoning downstream and within the context of our semantic-driven framework, we modify the traditional dimensional analysis approach to enable the mapping between the $\mathcal{X}$ and Π-spaces. Thus, we revisit and modify some of the definitions in

case-based reasoning [119] while preserving the important results of the Buckinghan's **Π**-theorem. This is done in three (plus one optional) steps:

(a) *Formal definition of rules.* Using the principle of cause and effect along with definition (#2) in [119], we define a rule as follows.

Definition 5.3.1. *(Rule)* A rule can be defined as a continuous function f that maps a set of premises $(x_1, ..., x_{n-1})$, to a conclusion x_n, with $x_i \in R^+$. The rule f is fired when all x_i are present with $i \in (1, .., n)$.

Thus, we write:

$$\Big[f(\underbrace{x_1, ..., x_{n-1}}_{premise(body)}) \quad \Rightarrow \quad \underbrace{x_n}_{conclusion(head)} \Big]_{(rule\ f)}$$

In the context of our framework and for the purpose of the dimensional analysis, the x_i are statements expressed as valid triples of the form *(*`NameClass`*,* `NameProperty`*,* `NameDatatype`*)*. They are constructed from elements of the TBox that are checked against corresponding instances in the knowledge base ABox at run time.

(b) *Definition of the similarity transform function π.* This function takes care of the transformation from the physical variables (with physical dimensions) space $\mathcal{X}$ to the dimensionless variables space Π. It is defined as follows.

Definition 5.3.2. *(Similarity transform function)* The similarity transform $\pi : \mathcal{X} \rightarrow \Pi$ is defined as

$$\pi_j = x_j \prod_{i=1}^{r} x_i^{-\alpha_{ji}} \qquad with \;\; j \in [1, m]$$

This represents a surjective mapping, since the similarity transform π of a space R^n into a space R^m with $m = n$ - r represents a dimensionality reduction. One key characteristic of this procedure is that different objects in $\mathcal{X}$ may be mapped onto the very same object in Π. The immediate consequence is that the inverse similarity transform $\pi^{-1} : \Pi \rightarrow \mathcal{X}$ results in a dimension expansion that cannot be unique.

In our framework, results of the surjective mapping are translated into triples in the system's rules engine. Triples in the $\mathcal{X}$ space, initially of the form (`PhysicalDomainNameClass`, `PhysicalDomainNameProperty`, `PhysicalQuantityNameDatatype`), are combined according to both system physics as well as model and system controls to formulate statements in the Π space with dimensionless datatypes. The new statements are of the form (`IntegratorNameClass`, `IntegratorNameProperty`, `DimensionlessNameDatatype`).

(c) *Invocation of the similarity transform F of f in Π-space.* The following corollary of the **Π**-theorem ensures that the similarity transform F of f holds [119].

Corollary 1.1. Let $f(x_1, ..., x_{n-1}) = x_n$ be the rule for all cases $p \rightarrow \infty$ with the premise $(x_1, ..., x_{n-1})_p$, then the similarity transform of the conclusion x_n of the rule f in form of

$$\left[F(\underbrace{\pi_1, ..., \pi_{m-1}}_{premise}) \quad \Rightarrow \quad \underbrace{\pi_m}_{conclusion} \right]_{(rule\ f\ \ similarity\, transform\ F)}$$

The most important consequence of this result for our framework is the guarantee of the validity and consistency of rules written in the Π space. In fact, X-space and Π-space related rules can be written and combined (if needed) in the system rules engine without sacrificing its expressiveness. By reducing the size of the decision

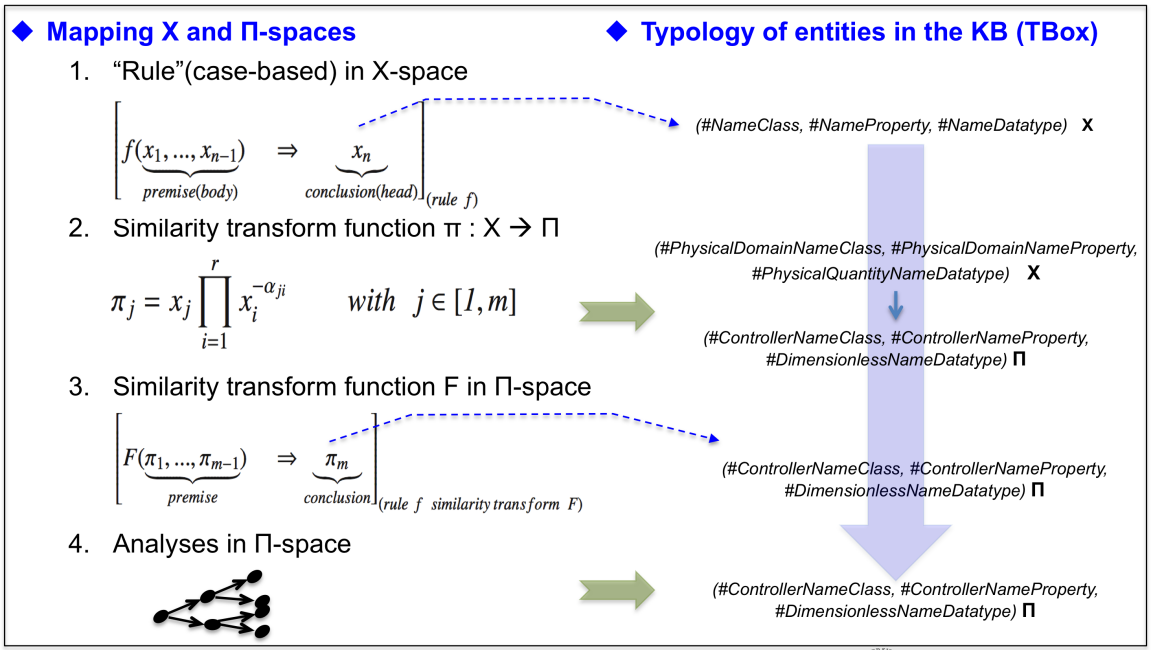


Figure 5.5: Modified dimensional analysis procedure for mapping between the physical space ($\mathcal{X}$) and dimensionless space(Π)

tree, the computational efficiency of the reasoning process is enhanced through fewer searches of the knowledge base for matches and fewer conflicts to resolve.

(d) *Extension of the analyses in the* Π*-space*. Rules, independently of the space in which they are written, can translate part (intermediary) or a full branch of the decision tree for the CPS safety problem at hand. Therefore, "rule-compatible decision trees" are inputs and outputs of the dimension analysis/reduction process. The experienced modeler/systems engineer can pursue the analysis directly in the Π-space - by means of generalization for instance - to fit the desired design goal. However, as we'll see in the case study in Section 5.4, the resulting insight gained in the analysis might come at the cost of having to deal with a larger decision tree.

Figure 5.5 summarizes the mapping process while showing the correspondence with the

entities in the knowledge base $\mathcal{K}\langle\mathcal{T},\mathcal{A}\rangle$ of a given (CPS) domain D as formally defined in Section 2.2.2.

5.4 Case Study: A Reasoning Framework for Traffic System Safety

5.4.1 Problem Description and Analysis

The purposes of this case study are twofold:

1. Show how the proposed CPS-KMoDS development chart in Figure 5.3 can be used to build the architecture in Figure 5.2 and,

2. Illustrate how the underlying reasoning structure can be used to support decision making and, consequently improve system level safety. Our focus will be on the domains layer of the architecture and its immediate parent (integration) and child (extension and computation support).

This case study considers the problem of self-driving cars approaching a traffic intersection controlled by a smart traffic light system [207, 209]. In this scenario, a vehicle (i.e., the physical system) interacts with the light (i.e., the cyber system) with the objective of maximizing performance subject to the constraint of "absolute" safety (i.e. vehicle crossings are safe at the intersection). The traditional master-slave relationship between the light and the vehicles are replaced with a cooperative relationship enabling a bidirectional communication between these entities. Thus, this system is an "ideal CPS" where each entity is equipped with its computation platform as per Figure 1.2. The problem domain and analysis procedure is formulated as shown on the left-hand side of Figure 5.3.

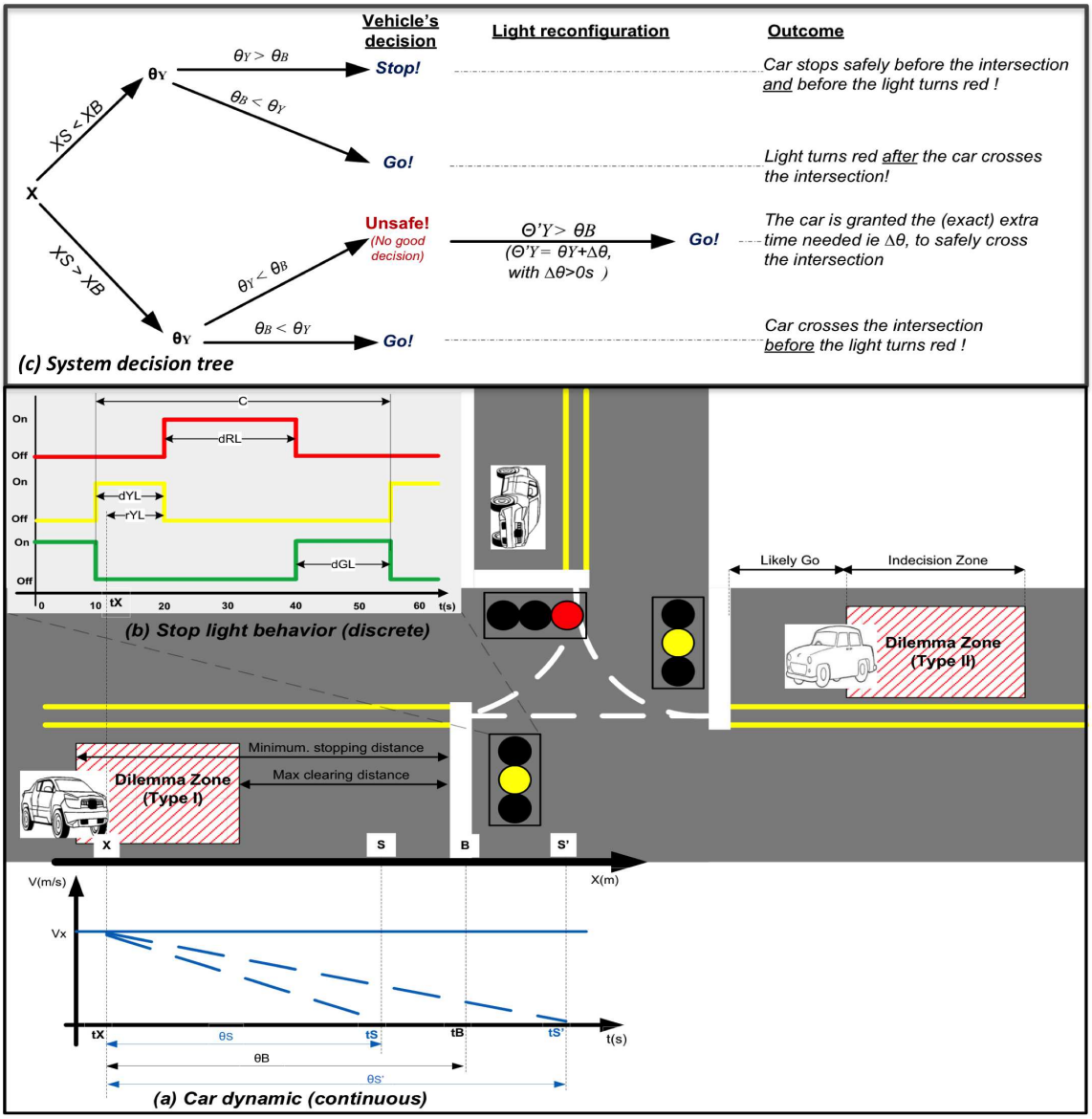


Figure 5.6: Schematic of spatial and temporal concerns in the dilemma zone problem. Traffic lights have discrete state behavior versus time. Here, C is the total cycle time for the lights. Variables dGL, dYL and dRL represent the duration of the green, yellow, and red lights, respectively. Variable rYL is the time remaining for the yellow light. Vehicles have dynamic behavior that varies continuously with time. Here, θS is the time it takes the vehicle to fully stop before the stop line, θB is the time to reach the intersection while traveling at speed V_x, and $\theta S'$ is the time it takes the vehicle to fully stop after the stop line.

Dilemma Zone: Definition and Existing Solution Approaches. Also called the twilight zone, Amber signal or decision zone, the dilemma zone is the area at a traffic intersection where drivers are indecisive on whether to stop or cross at the onset of a yellow light. Research [294] indicates that under such circumstances only 90% of drivers will "play it safe" and decide to stop. As result, the behavior of users in "twilight zones" claims around 2,000 lives and billions of dollars in damages at stop light intersections in the United States alone every year [130].

From an analysis standpoint (see Figure 5.6), scholars distinguish two types of dilemma zone that differ by the perspective adopted on the problem. *Type I* dilemma zone formulations (see center left side of the figure) place the "physics of the vehicle" at the center of the problem formulation and are concerned with the difference between the distance from the stop line to the nearest vehicle that can stop safely (i.e., minimum stopping distance) and the distance from the stop line of the farthest vehicle that can cross the intersection at the onset of the yellow light (i.e., maximum clearing distance) [131,166]. Therefore, the physical parameters of the situation (e.g., car speed, road and car conditions, and so forth) are the key determinants of whether the car will be able to safely cross the intersection or stop prior to the stop line. *Type II* dilemma zone formulations (see center right-hand side of Figure 5.6) are defined with regard to the driver's behavior and decision making as the vehicle approaches the intersection at the onset of a yellow light. The boundaries of this type of DZ are also sometimes measured with a temporal tag (i.e., representing the duration to the stop line) added to the probabilistic estimate [46]. In this work, we will adopt the Type I definition of the dilemma zone.

Past research has focused on finding ways to mitigate, or eliminate, DZs using mostly a pure traffic control engineering view of the problem. These efforts have resulted in signal timing adjustment solutions that ignore or cannot properly account for the physics of vehicles or driver's behaviors [174, 198, 301]. In order to deal with uncertainties, other scholars have used stochastic approaches such as fuzzy set [131] and Markov chains [166]. For all of these traditional techniques, the baseline of the solution can be either reduced (explicitly or not) to a space- or temporal-based dilemma zone, but not both.

For the purpose of this experiment, we keep the physics and representation of the vehicle simple and assume that: (i) the vehicle is a point at location (X) traveling at constant velocity V_X towards the light at location B as shown in bottom left hand side of Figure 5.6; (ii) entities execute actions as instructed in no delay. Also, (iii) computation and bidirectional communication are performed within actuation and sensing response time margin of error with no delay.

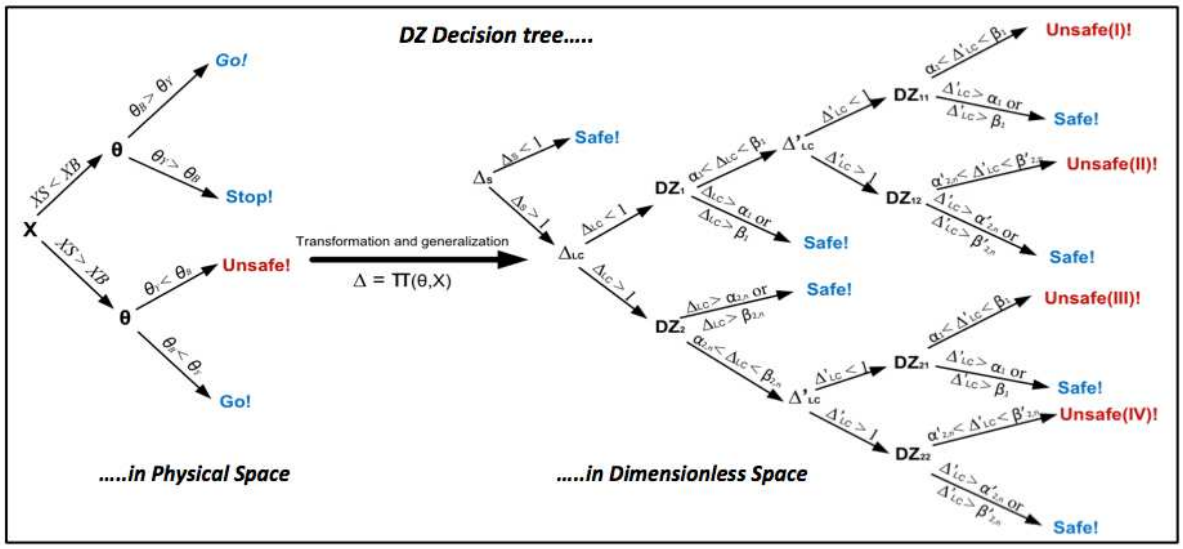


Figure 5.7: Framework for decision-making for the Dilemma Zone problem. Left: decision-making in the physical ($\mathcal{X}$) space. Right: decision-making in the dimensionless (Π) space.

Translating Safety Requirement Satisfaction into a Decision Problem. The core safety requirement of the system car-light that should be valid all the times is that *"No vehicle is allowed to cross the intersection when the light is red"*. This is a non-functional requirement, a hard constraint which violation is the driving force behind the multiple accidents observed at traffic intersections. As shown in Figure 5.6, the continuous dynamic of the vehicle (a) and discrete behavior of the light (b) illustrate the very different nature of both entities. This complicates the ability of the system to satisfy the safety requirement at the onset or in the presence of the yellow light. However, a deeper look of the problem shows that there is a way forward.

Understanding the mechanisms by which system-level safety is achieved or violated is critical in addressing the dilemma zone challenge. As shown in Section 5.3.6, decision trees are the most suitable analysis tool to explore the different possible paths the system could follow and characterize the resulting state as safe or unsafe. As shown in the left side of Figure 5.7, the probability of making the right course of action increases when the smart car has three key information at decision time: (1) Duration Θ_Y of the yellow light before it turns red, (2) Vehicle stopping distance XS and, (3) Travel duration Θ_B or distance to light XB. A smart car will be able to detect the light and accurately compute XS, Θ_B and XB on one hand, and take advantage of the bidirectional communication with the light to obtain Θ_Y from the stop light. Thus, it will be able to make a more informed decision as shown by the system decision tree on the left side of Figure 5.7. Although all paths of the decision tree do not lead to good decisions there is a way out, through the reconfiguration of the light (see Figure5.6(c)).

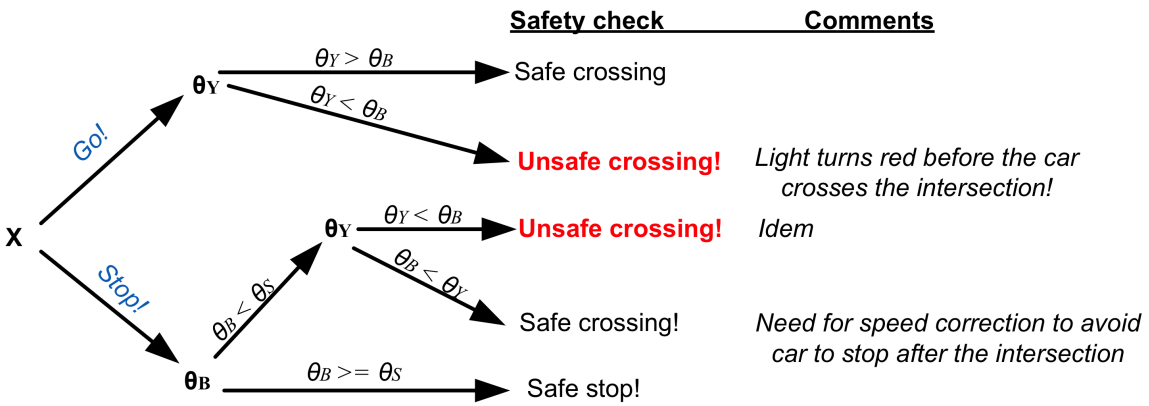


Figure 5.8: Decision tree for a human-driven car for the yellow light. Upon acknowledgment of the yellow light, the driver has to make a decision between keep going or stopping with approximate estimate of the travel distance to the stop line and no information on the time remaining on the duration of the yellow light. None of the 2 decision paths is 100% safe.

Compared to the decision tree of the human driver shown in Figure 5.8, this approach significantly improves system level safety and throughput by reducing the number of decision paths that lead to unsafe system states.

Reasoning Support to Prevent Unsafe System Configurations. The existence of a configuration of the system for which there is no good decision despite the car smartness highlights the prominent role of the physics in the overall system safety. One illustration is the situation where the speed and/or condition of the vehicle along with the one of the road do not allow it to stop safely before the stop light or cross it before it turns red. Thus, the system will enter an unsafe state, the vehicle physics preventing the safety requirement from being satisfied. In such situations, we make use of the bidirectional relationship and reasoning capabilities of both entities (and an intermediary traffic supervisory controller) to resolve this issue before it materializes. If the traffic light learns that a vehicle cannot possibly pass through the intersection safely, it will reconfigure its operations for instance by lengthening the duration of the yellow light by just the amount of time needed i.e.,

$\Delta\Theta$ (determined by the supervisory controller) for the car to cross safely. The additional time will be taken from the duration of the red light in the same cycle, making its total length unchanged. This will result in a safe crossing of the intersection as illustrated in Figure 5.6(c). The mechanics behind this reasoning process as well as the dimension transformation between the trees in Figure 5.7 will be discussed in Chapter 5.

5.4.2 Jena-based Modeling of the Traffic System: System Architecture

The implementation of the CPS-KMoDS architectural framework makes use of Semantic Web technologies introduced in Section 4.1. Semantic web technologies provide a variety of interfaces for accessing and handling standardized technologies such as RDF, triple stores and OWL platforms. Jena architectural framework supports the deployment of architectures that are consistent with the general architecture in Figure 5.2. That's what the construction process of the solution domain (as per Figure 5.3) of the dilemma zone problem does. The various layers of the CPS-KMoDS architecture in Figure 5.2 are individually implemented and programmatically assembled bottom up as per the architecture using the capabilities of Jena API. In the next sections we describe each layer of the CPS-KMoDS architecture for our traffic system example and their Jena-based assembly following the flow chart in Figure 5.3.

5.4.3 Domains Layer: Light, Car and Time Semantic Blocks

From a CPS perspective and as suggested by the architecture, our traffic system model is partitioned into subdomains. We keep the space domain simple (reduced to a

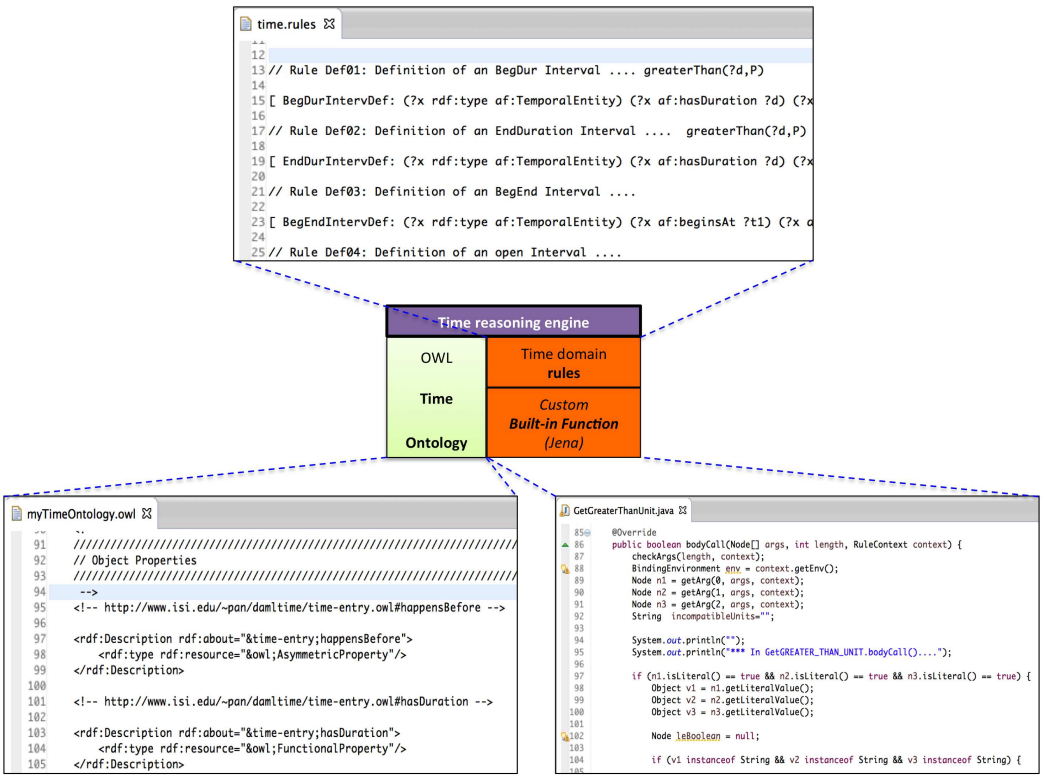


Figure 5.9: Time reasoning engine semantic block and its implementation.

point), thus, there is no need for a separate ontological description for this meta domain for this application. For each of the three foundational sub- domains (i.e., physical, meta and cyber), a corresponding domain specific ontology – car, time and light – is created along with domain rules. For instance, a car is defined in term of families such as `LightTruck`, `SUV` with properties like `hasWeight` and `hasFinalDriveRatio`. These properties are common to all types of car. Similarly, `hasColor` and `hasCycleDuration` are properties common to all stoplights. As for time, we employ a simplified version of the OWL-Time ontology (see Figure 3.2). Concepts such as `Instant`, `ProperTimeInterval` and properties like `beginsAt` and `intMeets` serve as the foundation for the domain. Extensions are programmatically added using the Jena ontology API. This results in the development of subsystem ontologies that provide a better definition of the subsystem for efficient future

use. For example, datatype properties such as `hasSpeed` and `hasStoppingDistance` are added to the car ontology because of their relevance to a formal quantification of the vehicle dynamics. This, in turn, is critical to the decision making strategies that solve the dilemma zone problem.

A set of rules is created for each domain-specific ontology and encoded in the corresponding rules engine. For example, the fragment of code:

```
// Rule #1: Propagate class hierarchy relationships ...

[ rdfs01: (?x rdfs:subClassOf ?y), notEqual(?x,?y)
-> [ (?a rdf:type ?y) <- (?a rdf:type ?x)] ]

// Rule #2: Infer an "Instant" from definition and property of temporal entity...

[ Instant: (?x rdf:type te:TemporalEntity) (?x te:hasTime ?t)
noValue(?x rdf:type te:Instant) -> (?x rdf:type te:Instant) ]

// Rule #3: Compute duration of a "Proper" time interval ...

[ GetDurationPropInterv: (?x rdf:type te:BegEndTimeInt) (?x te:beginsAt ?t1)
(?x te:endsAt ?t2) getDurInt(?t1,?t2,?d) noValue(?x te:hasDuration ?d)
-> (?x te:hasDuration ?d) ]
```

shows how the Jena rules engine relies on hybrid and forward chaining techniques (introduced in Section 5.3.5) respectively to propagate relationships among classes in a hierarchy (#1), define an entity (#2), and compute and infer new statements, possibly with the help of built-in functions (#3). Figure 5.9 shows the time reasoning engine semantic block and excerpts of the implementation of its various modules for our Dilemma Zone application.

5.4.4 Semantics Support Layer: Handling of Physical Quantities

The framework enables the branching of semantic extensions to domains ontological structures wherever it's needed. In the case of this application, there is a need

for our reasoner to properly handle physical quantities, dimensions (length and time) and units carried by data characterizing physical and meta properties such as `hasCarSpeed` and `hasDuration` in Car and Time ontologies. This is critical in keeping the reasoning and the ontologies consistent and unambiguous. Both flaws have the potential to lead to undecidable reasoning.

To that aim, we use the Jscience [5] package to capture and handle the representation, conversion and computation of physical quantities across the framework. This enables the reasoner to properly represent and distinguish, during processing and rules checking, both dimensions and units. These semantic services are provided to the reasoner by calls of Jscience functionality within custom built-ins functions where needed. Given the current inability of Jena to directly process dimensions and units, we wrap them into String datatypes as illustrated by the use of "XSDString" data type for the range of Car physical properties (see left-hand side of Figure 5.10).

With this step completed we can proceed to "local" testing of individual domain level as per Figure 5.3 by populating individual ontologies with valid instances. The verification of the proper integration of the Jscience and rules engine is of high interest here. A successful verification clears the path toward the integration of various blocks to form the integration layer for our traffic system.

5.4.5 Integration Layer: Integrator and System Level Reasoning

Traffic system Integrator. As shown in Figure 5.10, the traffic system integrator defines relationships between subdomain entities. It's a meta sub-domain of the traffic system that cross-cuts the various cyber, physical and other meta domain making up the system

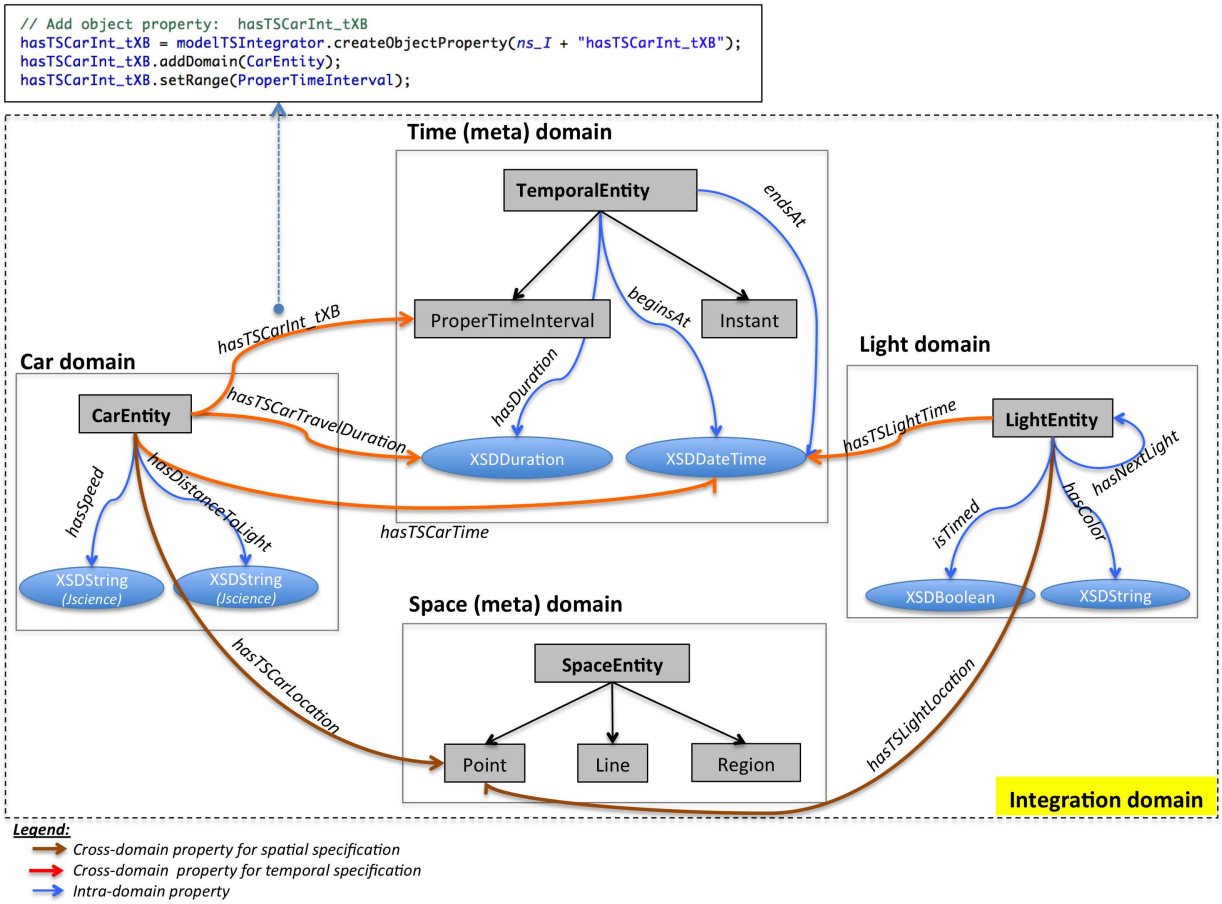


Figure 5.10: Illustration of the construction mechanism of the traffic system integrator ontology

and specifies cross-cutting system-level properties. This includes properties related to the metrics used to help characterize the decision space. Its a separate ontology that simply uses elements of subsystem ontologies to enable a system-level view of the traffic system. As a case in point, the excerpt in Figure 5.10 shows how Jena API is used to create and add a new ObjectProperty `hasTSCarInt_tXB` to the Integrator ontology using elements of the Car ontology (`CarEntity`) as domain and Time ontology (`ProperTimeInterval`) as range. This property defines and associates a proper (close) time interval to the period of time that a car travels from location X (when the decision is being made) to the stop light (location B). It is important to observe that the integrator operates like a traffic system

"semantic controller" with its own rules engine encoding and enforcing system-level rules and calculations that affect the domains involved in the CPS behavior i.e. car, time and light.

Traffic System: Model, Control and Reasoning Strategies. The efficient reasoning on the system, as a whole requires the integration of the various ontologies. The overall traffic-system model is constructed from the merging of individual ontologies, including the Integrator. We opt for a dynamic import of ontologies to manage the stream of data in the system. Thus, domain and integrator ontologies are added to the empty traffic system ontology as sub-models, with their top classes as disjoint subclasses of a `TrafficSystemEntity` class. A system-level TS rules engine is constructed by way of union of domain rules engines in a unique file with integrator rules serving both as controller and systems integration glue. Its configuration mirrors the various branches of the system decision tree. A predefined, generic Jena reasoner is used to perform inferencing because of its support for user-defined rules as well as forward, backward and hybrid chaining execution strategies. The integration of the units package Jscience with Jena thus as described in Section 5.4.4, enables the processing of physical quantities by the reasoner. This approach to construction of the TS model has the advantage of preserving the CPS view of the system while enabling deep insight in the connections and relationships between the domains. This is critical to uncovering and understanding mechanisms through which unsafe situations within the dilemma zone occur, while also providing support for efficient decision making.

Figure 5.11: Reconfiguration of the light to get the car out of an unsafe region.

5.4.6 Application Layer: Instantiation and Testing

In order to evaluate the effectiveness of the traffic system framework, we test it as a stand alone platform. We instantiate the ontological structure by populating the system with car and light entities and minimal data characterizing their basic properties. We are particularly interested in configurations of the system for which it reaches one of the four unsafe states. We verify that the reasoner is able to accurately: (1) predict this occurrence and, (2) reconfigure itself (actually the light) to enable safe crossing of the intersection when the car doesn't have a viable solution (NO_GOOD). To exercise the system, we pick

a `2004FordTaurusSES` (Sedan) weighing around 1.5 ton and approaching an intersection at 30 m/s. The remaining duration of the yellow light at the time the decision is taken is $r_{YL} = 9s$ on a total duration of $d_{YL} = 15s$. Combined with other parameters (e.g., stopping distance, braking force, other lights durations, etc...), the traffic system reasoner is able to infer that the vehicle system will enter an unsafe state, i.e. region IV (see right hand size of Figure 5.7). The screen capture in Figure 5.11 shows how the traffic system controller improves decision making in the dilemma zone by allocating extra time i.e., $\Delta\Theta$ = 2 s to the length of the yellow light, which is the time needed by the car to cross the intersection safely. The new system metrics are calculated to account for the change and ensure the integrity of the duration of the cycle of the stop light. Therefore, the car is no longer projected to violate the red light when it reaches the intersection, it's now in region V which is a safe spot in the decision space.

Chapter 6: Cyber-Physical Transportation Systems: Safety Metrics, Tubes and Analyses

6.1 Introduction

During the past three decades, transportation systems have been transformed by remarkable advances in sensing, computing, communications, and material technologies. The depth and breadth of these advances can be found in superior levels of automobile performance and new approaches to automobile design that are becoming increasing reliant on sensing, electronics, and computing to achieve target levels of functionality, performance and cost. By the end of this year, as much as 40% of an automobile's value will be embedded software and control related components [263,286]. Looking ahead, even greater levels of automation will be needed for self-driving cars [92,100]. While consumers applaud the benefits of these advances and the products they enable, engineers are faced with a multitude of challenges that are hindering the system-level development of cyber-physical transportation systems (CPTS). These challenges include:

1. The integration of CPS technologies into existing infrastructure,

2. The realization of "zero fatality" transportation systems, and

3. The development of formal models and credible, actionable performance and safety metrics [71].

To this end, metrics for system safety are needed to:

1. Evaluate the operation and control of transportation systems in a consistent and systematic way,

2. Identify, measure, and predict dynamic interactions among system components,

3. Set standards that serve as measure of effectiveness (MoEs) and guide MBSE efforts.

In this chapter, we introduce and describe a solution approach to these challenges through the development and simulation of metrics for safety analysis of CPTS. It builds on the lessons learned from the case study introduced in Chapter 5 to develop and simulate traffic systems safety metrics that help characterize and solve the dilemma zone problem. Thus, we consider the interplay among the key elements of transportation systems at traffic intersections, and their consequences on overall system level safety. The focus is on the development of metrics to capture the essence of these interactions, and support the characterization of the dilemma zone problem and its representation using three-dimensional dilemma tubes.

Section 6.2 introduces existing software technologies and infrastructure and ones under-development) used to support the implementation and simulation of the safety metrics and tubes. Section 6.3 discusses challenges in realizing cyber physical transportation systems and introduces the new dilemma zone metrics and their tubular representation. Section 6.4 describes the system architecture and simulation prototype of the dilemma tubes. Safety analyses are performed in Section 6.5.

6.2 Systems Integration and Simulation with Whistle

6.2.1 Whistle Scripting Language

The simulation and evaluation of CPS applications requires disciplined approaches to the integration and execution of models. We solve this problem with Whistle, a tiny scripting language designed for the integration and simulation of applications that are glued together. Among the key features of the language are its ability to: (1) support the use of physical units and dimensions from the problem description stage, (2) enable the use of variables, matrices, and looping and branching structures to control the flow of program logic and, (3) support the integration of custom-built functions (along with their names and arguments). The short fragment of Whistle code:

```
area     = 0.04 m^2;      // Cross section area of a pipe ...
velocity = 5 m/sec;       // Fluid velocity ....

print "*** Cross section area = ", area;
print "*** Fluid velocity     = ", velocity;
print "*** Discharge rate     = ", area*velocity;
```

shows, for example, computation of the flow-rate through a pipe. Notice how the physical units are built right into the language! For a detailed description of the language capabilities, see Delgoshaei, Austin, and Pertzborn [65].

Whistle is implemented in Java. As such, its computational support interface enables the scripting language to handle input and output of model data from/to files in various formats (XML, Open Street Map (OSM), Java, etc.). Figure 6.1 shows, for example, visualization of layers of data – buildings, runways, service roads, etc – associated with Baltimore-Washington International (BWI) Airport. Behind the scenes, Open Street

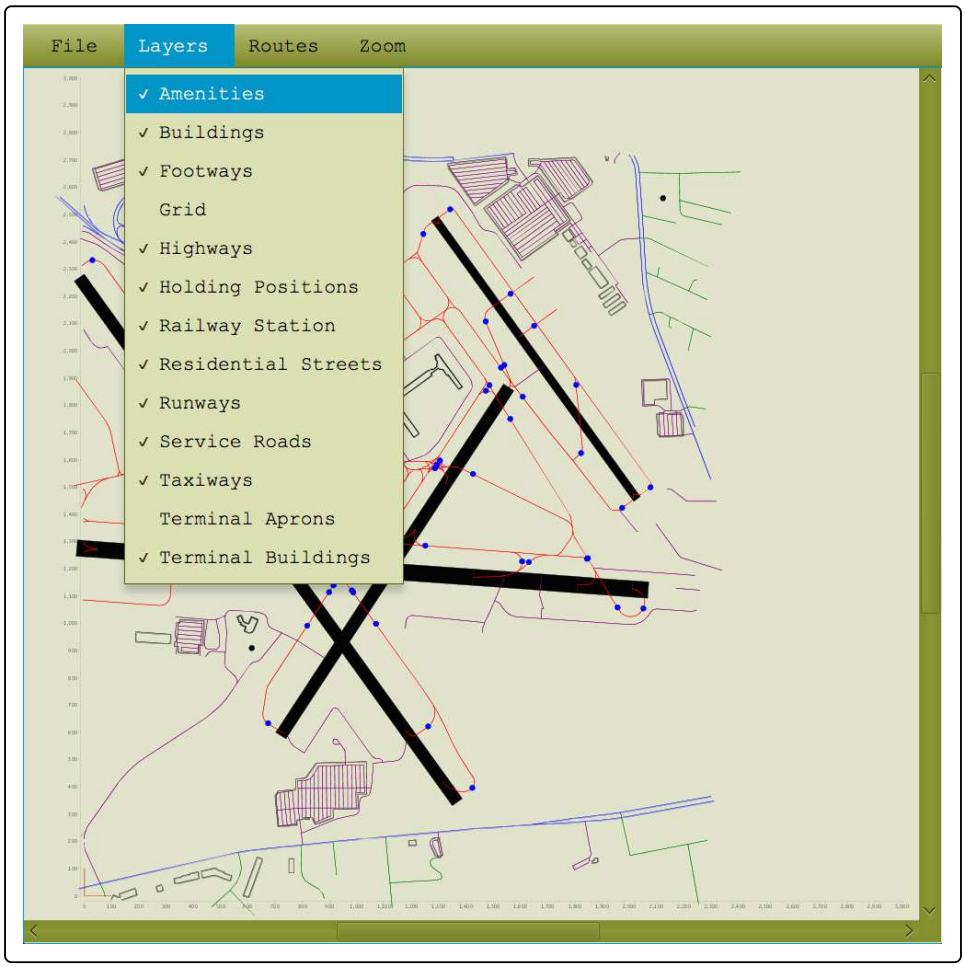


Figure 6.1: Visualization of Open Street Map data in Whistle. This example shows layers of data – buildings, runways, service roads, etc – associated with Baltimore-Washington International (BWI) Airport.

Map data is imported into Whistle and stored as an Open Street Map (OSM) Model. Then, layers of discipline-specific data are systematically extracted from the OSM model, stored as workspace composite hierarchies, and added to the JavaFX visualization model. Composite hierarchies are multi-layer tree structures of arbitrary complexity, and are implemented in a flexible and scalable manner via the composite hierarchy software design pattern [90]. See Figure 6.2.

Whistle also makes extensive use of the model-view-controller (MVC) software design pattern. As illustrated in Figure 6.3, MVC provides a clear separation between

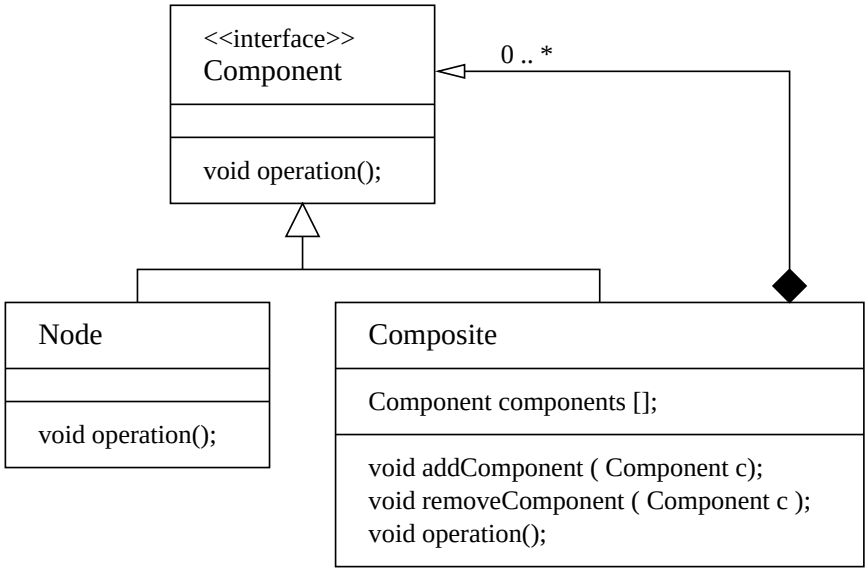


Figure 6.2: Composite class diagram.

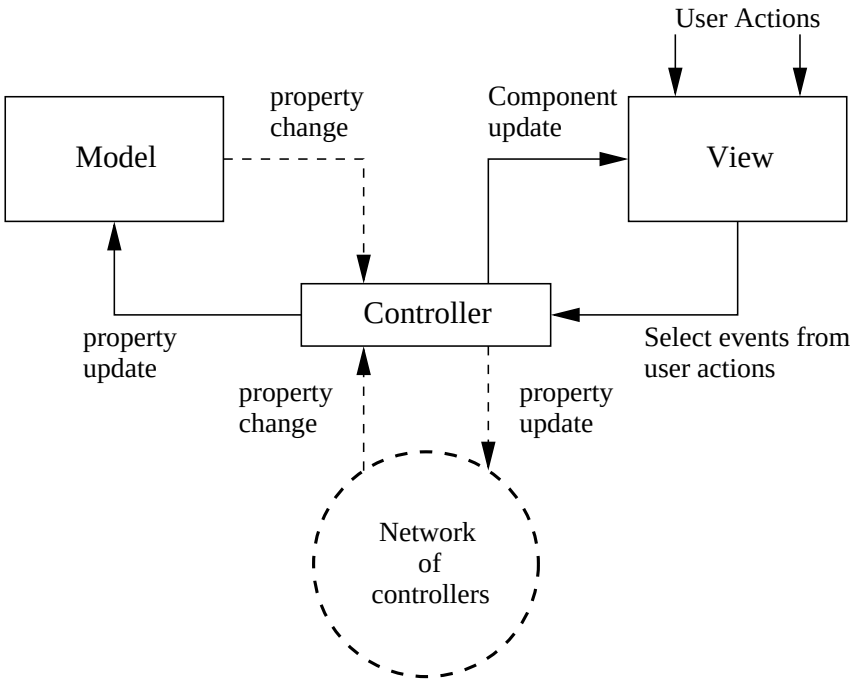


Figure 6.3: Implementation of model-view-controller with the control acting as a mediator.

domain and presentation objects, and enables communication among multiple models and views. Notice that there is no direct link between the views and models. Instead, all exchanges go through the controller located at the center of the pattern (which could include other controllers), to process and route all the communications between models and views. Changes in the model are propagated to registered controller(s) which updates the view(s) accordingly. Controllers also update model properties in response to notifications by the view of some user action. Thus, overall, the controller plays the role of a mediator in the communications between models and views.

6.2.2 Systems Integration with Whistle

Figure 6.1 shows layers of geographical data for BWI airport. This data is obtained from Open Street Map and it is static. The next step in Whistle capability is computational support for the simulation of behaviors in CPS applications.

As illustrated in Figure 6.4, this will require the scripting of problem solving strategies that drive behaviors, but evaluate them with respect to metrics involving time and space. Additional visualizations, such as statechart behaviors, will support for synchronized data/information in models and views, and across concurrent processes. There is also a need for traceability mechanisms that properly link discipline-specific domains, and across various stages of system development (e.g., requirements, design, simulation, operation, etc.).

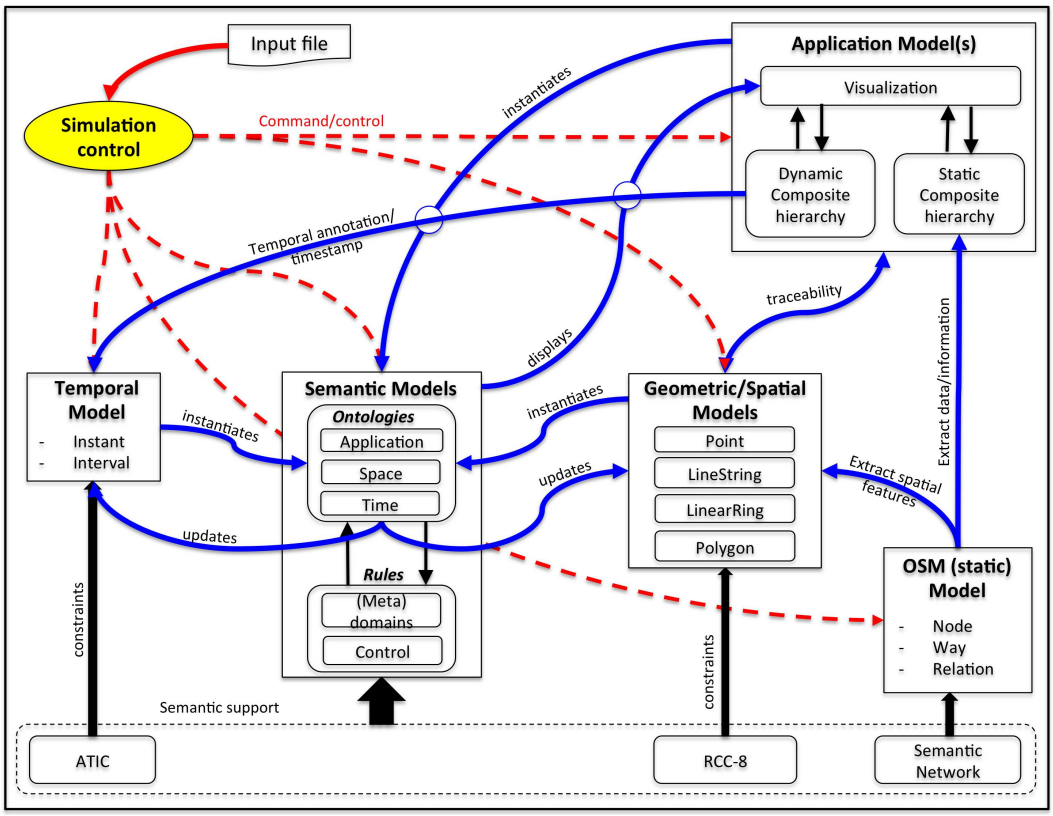


Figure 6.4: Simulation architecture for spatio-temporal reasoning.

6.3 Safe CPTS: Metrics for Characterizing the Dilemma Zone Problem

6.3.1 Cyber-Physicality of Traffic Systems

CPTS development challenges and need for metrics. During the past three decades, transportation systems have been transformed by remarkable advances in sensing, computing, communications, and material technologies. The depth and breadth of these advances can be found in superior levels of automobile performance and new approaches to automobile design that are becoming increasing reliant on sensing, electronics, and computing to achieve target levels of functionality, performance and cost. As of now (2016), as much as 40% of an automobile's value is embedded software and control related components [263, 286]. Looking ahead, even greater levels of automation will be needed for self-driving cars [92, 100].

While consumers applaud the benefits of these advances and the products they enable, engineers are faced with a multitude of challenges that are hindering the system-level development of cyber-physical transportation systems (CPTS). These challenges include (a) the integration of CPS technologies into existing infrastructure, (b) the realization of "zero fatality" transportation systems, and (c) the development of formal models and credible, actionable performance and safety metrics [71]. To this end, metrics for system safety are needed to (1) evaluate the operation and control of transportation systems in a consistent and systematic way, (2) identify, measure, and predict dynamic interactions among system components, (3) set standards that serve as measure of effectiveness (MoEs) and can guide MBSE efforts. The continue high death toll at traffic intersections reminds

us that despite these advances there still lot of work to do to tackle these challenges.

Autonomous Cars and Intelligent Traffic Control Systems. Recent work [3, 280] illustrates the switch of researchers' interest toward investigating solutions to the DZ problem that incorporate both the car physics and light timing, while also providing a pathway forward for vehicle-to-infrastructure (V2I) interactions and integration. These solutions will soon become a reality, in part, because of an increased use of artificial intelligence in automating the command and operation of both cars and traffic signals. For automobiles, many aspects of autonomy – from braking to cruise control and driving functions – are in advanced stages of experimentation. Finding ways to put smartness into vehicles has contributed to reduced fatalities on highways mostly in the developed world. The enhancement of traffic signal controls with artificial intelligence is an idea whose time has arrived – indeed, we now have the capability to determine the position, speed and direction of vehicles, and adjust light cycling times in a coordinated way to make the intersection crossing more efficient. Researchers have been developing and testing various technologies with mixed results [44, 133, 241]. As a case in point, a pilot study conducted by Carnegie Mellon University, reports a 40% reduction of intersection waiting times, an estimated 26% decrease in travel time, and a projected 21% decrease of CO_2 emissions [44]. Tapping into the full potential of these intelligence capabilities is hindered by practical constraints that include (1) most vehicles cannot currently communicate with traffic light controllers, and (2) autonomous vehicles still struggle in operating safely in adverse weather conditions (heavy rain, snow covered roads, etc.) and changing environment (temporary traffic signals, potholes, human behaviors, etc.). In this work, we assume that these problems will be resolved by ongoing research activities.

Toward Cyber-Physical Traffic Management Systems. Real-time situational awareness (e.g., traffic, location, speed) and decision, combined with vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communications and control are valid and effective pathways for a solution to both congestion and safety at intersections. As such, we fully adopt a CPS view of the traffic system with regard to the DZ problem. The value of this perspective has already been demonstrated in Section 5.4. Autonomous vehicles (i.e., the physical system) interact with the light (i.e., the cyber system) with the objective of maximizing traffic throughput, while ensuring vehicle crossings are safe at the intersection. Enhanced performance and safety at the intersection have been proven possible, thanks to the critical role of temporal semantics in improving system level decision-making. Also, when bi-directional connections between the vehicle and light are possible, new relationships can be established to characterize their tight coupling – this, in turn, enables the various computers in the CPTS to exchange information, reason, and make informed decisions. These capabilities become safety-critical for situations – hopefully, rare situations – where behavior/physics of a vehicle is such that they can neither stop, nor proceed, without entering and occupying the intersection while the traffic light is red. Therefore, the development of metrics for the DZ problem will greatly benefit from and enrich the CPTS perspective.

6.3.2 Metrics for Characterizing the Dilemma Zone Problem

From Decision Trees to Dilemma Metrics. In Chapter 5, we've seen mechanisms through which safety requirements can be translated into decision trees in the physical domain. Moving forward requires a deep understanding of the interrelationships between cross-cutting system parameters from the various domains (car, light, time, space) involved

at meta level, especially the temporal one. Also, the ability of the system to efficiently reason about unsafe situations and find a satisfactory way out is critical. Learning from the benefits of DA (see Section 5.3.6), we argue that this complexity can be kept in check by casting the problem in dimensionless terms. Thus, we define and set up a transformation

$$\Delta = \Pi(\Theta, X), \tag{6.1}$$

of the initial decision tree from the physical space to a dimensionless space. The dimensional analysis for reasoning optimization procedure introduced in Section 5.3.6 guarantees the consistency of our results in both the physical ($\mathcal{X}$) and dimensionless (Π) spaces. Expressing the system decision tree in dimensionless space as a result of the transformation Π necessitates the definition of intermediary variables and parameters.

We begin by noting that the car will not always catch the onset of the yellow light; thus, what is really relevant for efficient decision-making here is the time left before the stop light turns red. Using the remaining duration of the yellow light r_{YL}, its full duration d_{YL} and the ones of the green and red lights i.e., d_{GL} and d_{RL}, we define the duration of a stop light cycle C, reduced cycle C_{YL} and cycle index k as follows.

$$C = d_{YL} + d_{RL} + d_{GL} \tag{6.2}$$

$$C_{YL} = r_{YL} + d_{RL} + d_{GL} \tag{6.3}$$

$$k = \frac{C}{C_{YL}} \tag{6.4}$$

$$\tag{6.5}$$

The short (α_1) and full (α_2) yellow light duration as well as the short (β_1) and full (β_2) stop light indexes are defined as follows.

$$\alpha_1 = \frac{r_{YL}}{C_{YL}} \tag{6.6}$$

$$\alpha_2 = \frac{d_{YL}}{C_{YL}} \tag{6.7}$$

$$\beta_1 = \frac{r_{YL} + d_{RL}}{C_{YL}} \tag{6.8}$$

$$\beta_2 = \frac{d_{YL} + d_{RL}}{C_{YL}}. \tag{6.9}$$

We add to the aforementioned physical variables the stopping duration $\Theta_B^{'}$ of the car – should it decide to stop – and define the **car stopping distance metric** Δ_S, the **light-car crossing time metric** Δ_{LC} and the **light-car stopping time metric** $\Delta_{LC}^{'}$ as follows.

$$\Delta_S = \frac{XS}{XB} \tag{6.10}$$

$$\Delta_{LC} = \frac{\Theta_B}{C_{YL}} \tag{6.11}$$

$$\Delta_{LC}^{'} = \frac{\Theta_B^{'}}{C_{YL}}. \tag{6.12}$$

All these metrics are dimensionless and serve as the key decision points of the dimensionless decision tree shown on the right-hand side of Figure 5.7. Literally, the *car stopping distance metric* captures the percentage of the allowed travel distance the car will need to cover to stop safely (if it can). The light-car crossing time metric measures the percentage of the light reduced cycle duration needed by the vehicle to arrive safely at the location B of the

stoplight (i.e. the stop line) while traveling normally. Finally, the light-car stopping time metric determines the percentage of the light reduced cycle duration to be covered by the vehicle at arrival at the stoplight while braking.

Navigating the Decision Tree. Navigation of the decision tree is facilitated by the equation pair

$$n = E\left(\frac{\Delta_{LC} - 1}{k}\right) \tag{6.13}$$

$$n^{'} = E\left(\frac{\Delta^{'}_{LC} - 1}{k}\right) \tag{6.14}$$

We employ the integer part function E to define indexes n and $n^{'}$. Equations (6.13) and (6.14) simplify the definition of α and β indexes when $\Delta_{LC} > 1$ or $\Delta^{'}_{LC} > 1$ as follows.

$$\alpha_{2,n} = k * \alpha_2 + k * n + 1 \tag{6.15}$$

$$\beta_{2,n} = k * \beta_2 + k * n + 1 \tag{6.16}$$

$$\alpha^{'}_{2,n} = k * \alpha_2 + k * n^{'} + 1 \tag{6.17}$$

$$\beta^{'}_{2,n} = k * \beta_2 + k * n^{'} + 1 \tag{6.18}$$

Along with equations (6.6) through (6.9), the values of α and β (see equations (6.15) through (6.18)) are necessary and sufficient to constrain the dimensionless metrics Δ_S, Δ_{LC} and $\Delta^{'}_{LC}$ and render a complete view of all possible outcomes of the decision tree in a dimensionless space Δ. From the right-hand side of Figure 5.7, we can see that there are four possible configurations of the system for which it is unsafe.

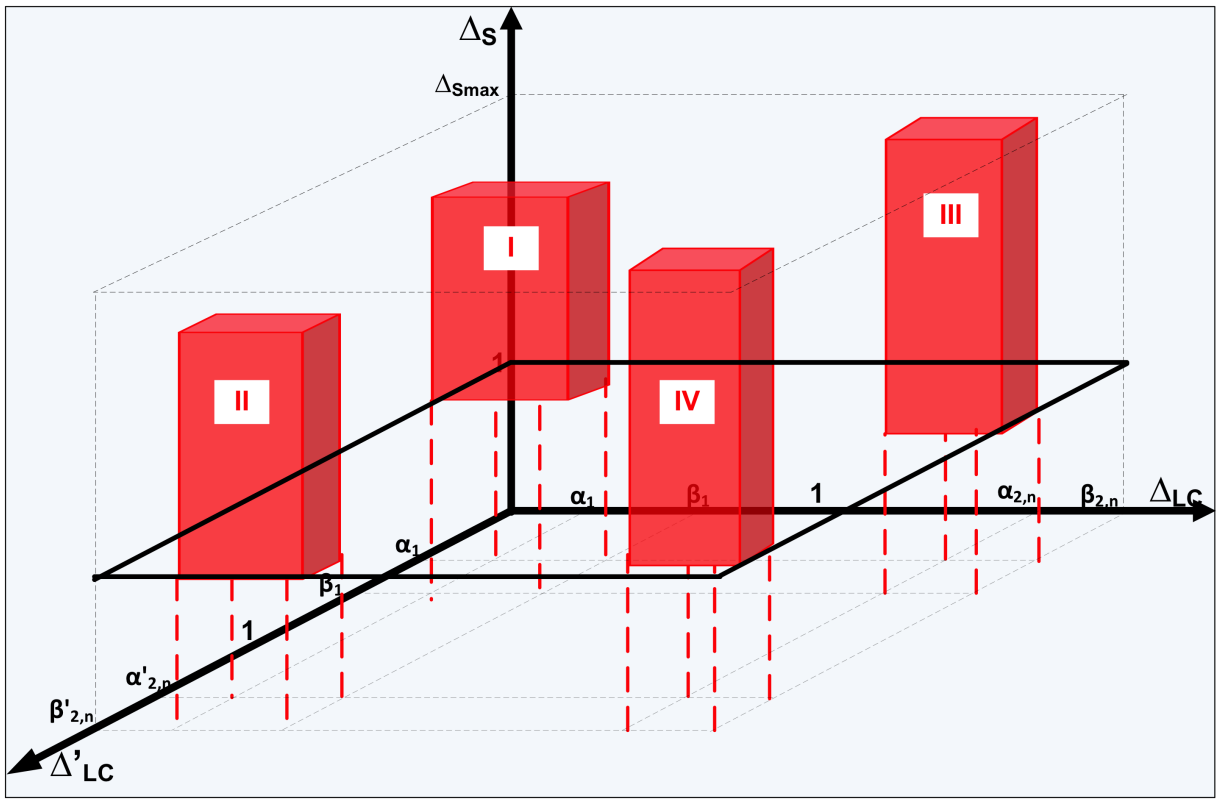


Figure 6.5: Dilemma tubes in the dimensionless (Δ) space.

From Dilemma Metrics to Dilemma Tubes. Each of the decision tree pathways on the right-hand side of Figure 5.7 that leads to an unsafe system state can be represented as a "dilemma tube" in the Δ space, as shown in Figure 6.5. For instance, equations (6.6), (6.8), and (6.10) through (6.12) provide the foundational elements for defining Tube I. The boundaries of each of the four tubes (i.e., I, II, III and IV) correspond to the above-mentioned parameters, with the maximum value of Δ_S i.e., Δ_{Smax} corresponding to the maximum value of all the Δ_S values in the system. Physically, this is determined by the physics of the family of vehicles crossing the intersection and the configuration of the traffic intersection as captured by equation (6.10). If, at any point in time, the system is projected to enter an unsafe state, this situation will be materialized as a point coordinate

$P_{\Delta}(\Delta_S, \Delta_{LC}, \Delta'_{LC})$ that is located inside a particular tube. The physical interpretation of such phenomenon is that the autonomous car does not have a good decision option, and will need external (light) help to safely cross the intersection.

Scenarios that lead to unsafe system configurations (e.g., see the right-hand side of Figure 5.7) will follow branches of the decision tree that terminate with an "Unsafe" system state. While the actual behaviors might not evolve along the pathways presented in the decision tree, the end result will invariably be the same (i.e., the system will be projected to enter an unsafe state). In practice, simulation and safety calculations can be done concurrently and the location of the resulting point coordinate relative to any of the four dilemma tube types easily determined. A final important point to note is that since each of the tubes is mutually exclusive, a vehicle can only be in one of the four dilemma tubes at a time, or in any location in the remaining part of the Δ space, i.e., a safe region. Knowing in which tube the unsafe state has been materialized is critical in determining the appropriate course of action to prevent the occurrence of an accident.

6.4 System Architecture and Implementation

This section introduces a Java-based software system infrastructure that adheres to the CPTS perspective and supports the tube framework described in Section 6.3.

6.4.1 System Architecture

It makes extensive use of the the MVC design pattern introduced in Section 6.2.1 to create and integrate models (of components and tubes) with simulation views (tubes and traffic system) glued together by an integration platform that acts as a controller. A

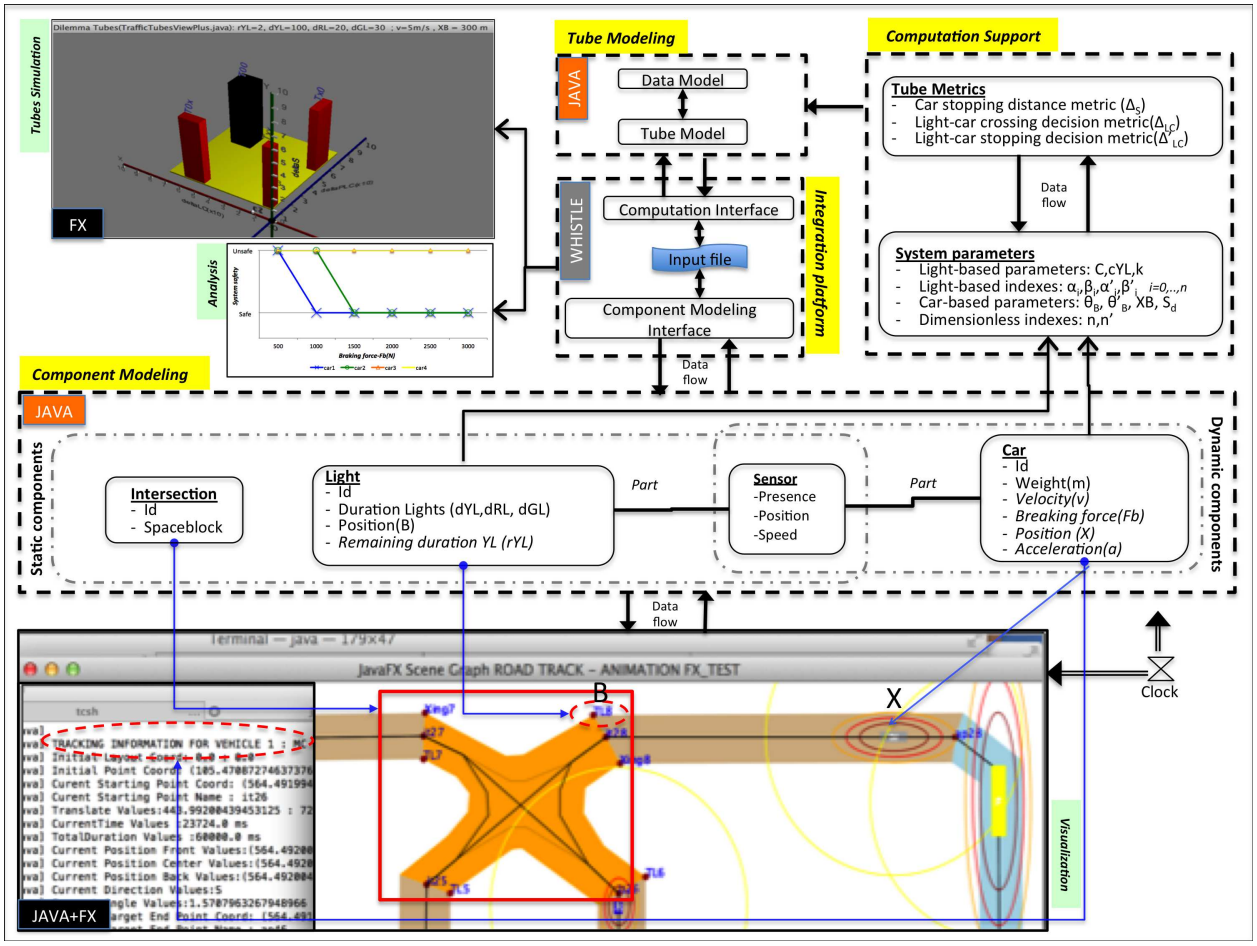


Figure 6.6: Dilemma tubes simulation system architecture. The latter follows the MVC design pattern and integrates models (of components and tubes) with simulation views (tubes and analysis) glued together by an integration platform that acts as a controller. A computation platform provides support for the calculations and ensures data consistency across domains. The system architecture is augmented with workspaces for traffic intersection simulation

computation platform provides support for the calculations and ensures data consistency across domains as shown in Figure 6.6. The system architecture contains workspaces for traffic intersection simulation and its main modules are as follows.

1. Component Modeling. The component modeling module plays a central role in the system simulation. Physical entity models are organized into static and dynamic components, as shown in the mid-section of Figure 6.6. Examples of the former include the traffic intersection (i.e., the spatial boundary), traffic lights, and their associated sensors. Their key attributes are not expected to change over time such as the stop light durations d_{YL}, d_{RL} and d_{GL} for the yellow, red and green for each cycle. The remaining duration of the yellow light (r_{YL}) is a key attribute of interest for our study that does decrease with time. As such, the component modeling module needs a clock to account for the elapsed time. In our formulation, sensors play a key role in determining the location (X) and velocity (v) of a vehicle as a function of time. With X and v in place, vehicle accelerations can be computed from the underlying equations of motion. Also, the vehicle braking force (F_b) is subject to change over time; thus, it is a variable of the system.

2. Tube Modeling and Metrics Computation Support. DZ tubes are modeled as software entities because they are not physical entities. In order to properly account for the multiple facets of tubes in this framework, and provide flexibility in the architecture, we propose that tube models serve as a data repository platform and bridge between the computation and the integration modules (see the dashed boxes and connecting arrows in Figure 6.6).

The interface for the data repository platform distinguishes *base tubes* (not visualized) from *dilemma tubes*. The former store the basic initial configuration of the stop

light, and information that will be used to create the latter (i.e., dilemma tubes). Dilemma tubes of various types allow for the representation of unsafe system states as defined by the car stopping distance metric Δ_S, the light-car crossing time metric Δ_{LC}, and the light-car stopping time metric Δ'_{LC} and specifications in equations (6.4) thru (6.18). This separation of concerns provides modularity and flexibility to the architecture, enabling the support for modeling of complex intersections with multiple stop lights on multi-lanes and/or complex intersection configurations (T,Y,X, etc.).

The visualization system interface (not shown) connects with the integration module, thereby allowing for flows of data to/from the visualization display, and in accordance with the adopted GUI technology. In our software prototype (see the top left-hand corner of Figure 6.6), the display is controlled from the integration module. On the interface with the computation support module, a *traffic tube* model is created as an extension of a more basic tube model. It is the ultimate data structure of the tube as it links predefined and computed tubes variables. The initial traffic tube is linked to the base tube, and dilemma tubes are created from updates of corresponding traffic tubes for various values of r_{YL}. The number of dilemma tubes to be visualized is computed by the system, based on values of n and n' as defined by equations (6.13) and (6.14).

The computation support module enables the correct calculation of the various metrics and variables needed to efficiently characterize the dilemma zone using the tube framework. It receives input data from both the component and the tube modules, processes computation request using equations (6.2) thru (6.18). We distinguish *system parameters* from the three *tube metrics* Δ_S, Δ_{LC}, Δ'_{LC} introduced above. The former are computed car, light or dimension parameters and indexes that will contribute in the

computation of the latter. Dimensionless indexes are parameters as they are, by definition, dependent on Δ_{LC} and Δ'_{LC}. Most of these parameters are defined as attributes of the traffic tube model thus, the results are stored as per the specification of that data structure.

3. System Integration. Reaping the benefits of the system architecture requires bringing together its various modules and pieces in an organized but systematic way. Thus, we need a way to assemble system models for the purpose of the various analysis needs. We solve this problem with Whistle. Currently, computational support is added, enabling Whistle to handle input and output of model data from/to files in various formats (XML, OSM, Java, etc.). Therefore, an input file (containing any Whistle-compliant program) is an integral and central part of this module. It provides access to other system modules and needed functionality via interfaces encoded as scripts. Also, the sequencing and timing in the execution of the commands is encoded in the program, giving the analyst/modeler the control of the execution of the simulation.

6.4.2 Simulation Prototype

We describe in this section an implementation of the framework for a scenario where the system configuration leads to a system state inside Tube I, as shown in Figure 6.5. The implementation consists of step-by-step assembly of a (typical) dilemma zone scenario, simulation, and analysis of the results. It is subject to three simplifying assumptions: (A1) the air resistance is negligible, (A2) there is a two-way, delay-free communication between the light and the autonomous car, and (A3) computation and reaction times are negligible.

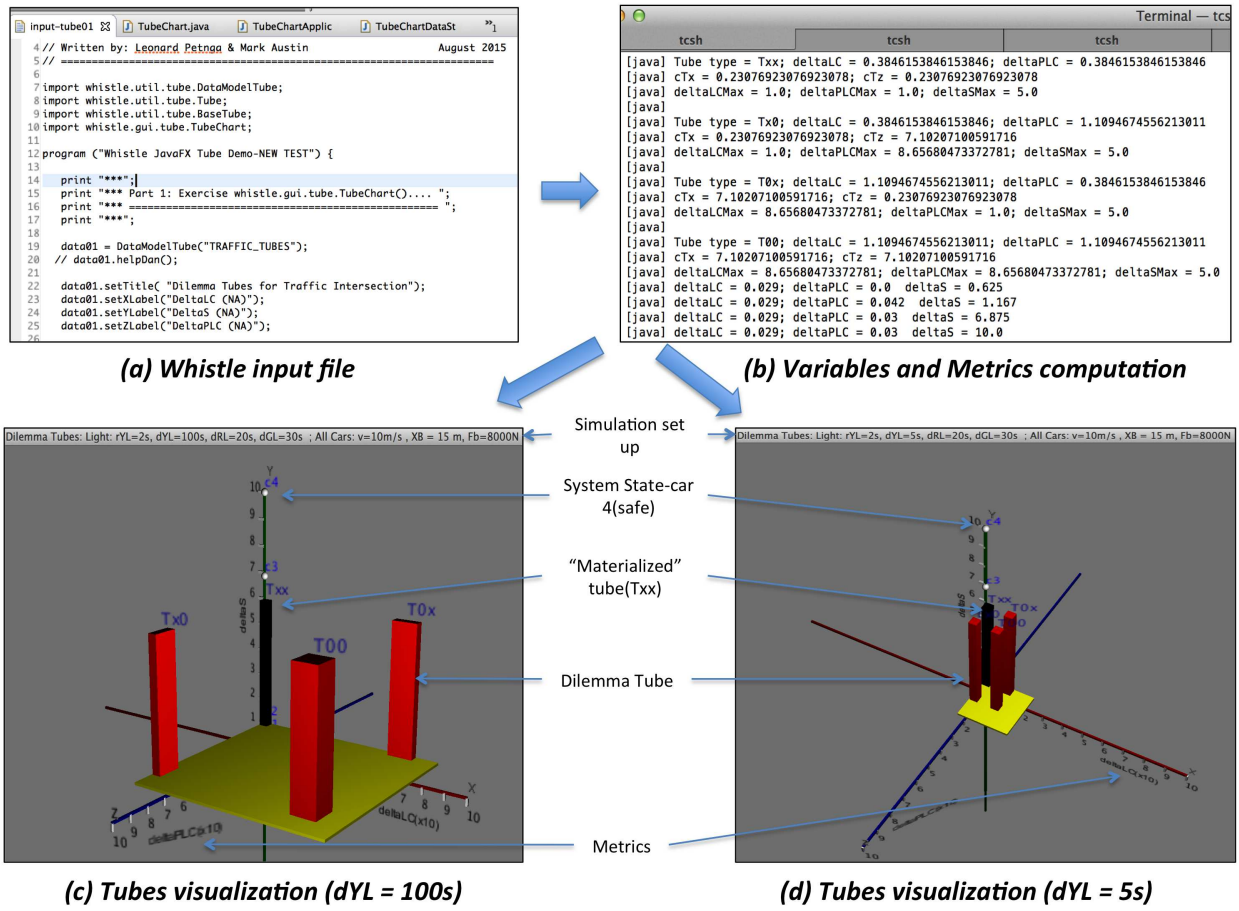


Figure 6.7: Schematic of system inputs and outputs. The sub-figures are: (a) Whistle input file, (b) variables and metrics computation, (c) tubes visualization for dYL = 100 seconds, and (d) tubes visualization for dYL = 5 seconds.

1. Step-by-Step Assembly of a Real-World Scenario. The step-by-step details are as follows.

(i) A traffic system controller of a smart traffic system computes and stores in real-time each stop light indexes (C, C_{YL}, k, α_i, β_i, i=1,2) based on its corresponding parameters (r_{YL}, d_{GL}, d_{YL}, d_{RL}) using equations (6.2) through (6.12).

(ii) An autonomous car approaching the intersection at speed s is given its distance XB to the stop line in real-time. This information is provided either by its on-board radar coupled with its computer or by the intersection controller. The car itself (autonomous

vehicle equipped with camera) notices the onset (or the presence) of the yellow light.

(iii) Based on its current acceleration, speed, road conditions, and maximum applicable braking force, the on-board computer of the car estimates the vehicles stopping distance XS, and computes Δ_S using equation (6.10).

(iv) The computer finds that $\Delta_S > 1$, meaning the car cannot be safely immobilized before the stop line. It then determines the normal travel time θ_B to go through the intersection, i.e., to cover the distance XB, should it decides to go at speed s.

(v) The car requests and obtains from the traffic controller the values of α_i, β_i, i=1,2 and the length of the reduced cycle C_{YL}. It then computes the light-car crossing metric Δ_{LC} using equation (6.11).

(vi) The on-board computer finds that $\alpha_1 < \Delta_{LC} < \beta_1$. At this point, the only way for the car to avoid violating the safety requirement (i.e., never cross the stop line when the light is red) is to hope that while braking, it will cross the stop line when the line is still yellow.

(vii) Using equation (6.12), the car determines the travel time θ'_B to cover the distance XB while stopping. Then, it computes the light-car stopping time metric Δ'_{LC}.

(viii) The on-board computer finds that $\alpha_1 < \Delta'_{LC} < \beta_1$, which translates as the light will be already red when the car crosses the stop line while stopping.

Individual values of the metrics Δ_S, Δ_{LC} and Δ'_{LC} generate a point coordinate somewhere within the dilemma Tube I, as pictured in Figure 6.5. The physical interpretation of this system state is that the vehicle does not have a good decision option, and will need a change of course of action or help from the light to safely cross the intersection.

2. Simulation Setup and Coverage. The simulation setup relies extensively on Java and its advanced graphics and media packages JavaFX as supportive technologies to create,

Element	Variable	Unit	Min	Max	Set value	Predefined parameters
Car	XB	m	10	60	30	m_1=1,500 kg, m_2=2,800 kg,
	F_b	N	3000	8000	5000	m_3=16,500 kg,
	v	m/s	5	30	10	m_4=24,000 kg
Light	r_{YL}	s	0	5	2	d_{RL}=20s
	d_{YL}	s	3	17	5	d_{GL} = 30s

Table 6.1: Summary of simulation parameters.

test, debug, and deploy a client application. Simulation coverage consists of four cars c_i, $i \in \{1, 2, 3, 4\}$ of different size (sedan, SUV, bus, cargo truck) and a stop light. Vehicles will be distinguished by their weight (m). Vehicle velocity (v), braking force (F_b) and distance to stop light line (XB) are discrete parameters that can be selected within a predefined range by the modeler/analyst. As for the stop light, the duration of the red light (d_{RL}) and green light (d_{GL}) are treated as constants; the duration of the yellow light (d_{YL}) and the corresponding remaining duration (r_{YL}) are discrete variables within predefined range. The range of each parameter is generally distributed around an average value that is used when a fixed value for a specific parameter is needed. Table 6.1 summarizes the case vehicles and parameter values employed in this simulation.

3. Simulation Execution and Dilemma Tubes Visualization. Visualization of the dilemma tubes occurs through a processing pipeline that involves the acquisition, storage, processing, flow and restitution of data between the input file and the visualization platform. For the execution of a scenario involving one car and one stop light, the following steps will be completed.

(1) A user creates an input file containing an execution/simulation program in a Whistle-compliant format. In this application we use a text file, such as the one shown in Figure 6.7(a).

(2) The program instantiates a tube `DataModel` customized to the needs of the simulation. This will later serve as a place holder for the various versions of tubes as they are constructed and displayed.

(3) The system is initialized. This is done by configuring the stop light with predefined values to d_{YL}, d_{RL} and d_{GL}. As for the car, if the engineering simulation module (e.g., racetrack) is hooked to the integration platform, then a car type is selected based upon its weight and its physical parameters (initial velocity, trajectory and position). The corresponding component models are interfaced with the integration module.

Computational requirements during the simulation can be reduced through pre-computation and storage of the dilemma tube parameters, as described in the following steps (4)-(7). This is done for various values of r_{YL} and dimensionless indexes n and n' (see equations (6.13) and (6.14)).

(4) The number of dilemma tubes N that need to be visualized at each iteration of r_{YL} is determined as follows.

$$N = \begin{cases} 1 & \text{if } n \text{ and } n^{'} \text{ are undefined} \\ n+2 & \text{if } n \geq 0 \text{ and } n^{'} \text{ undefined} \\ n^{'}+2 & \text{if } n \text{ undefined and } n^{'} \geq 0 \\ (n+2)(n^{'}+2) & \text{if } n^{'} \geq 0 \text{ and } n \geq 0 \end{cases} \tag{6.19}$$

In equation (6.19), n is undefined when $\Delta_{LC} < 1$ and n' is undefined when $\Delta'_{LC} < 1$. In this configuration, the only tubes that can be viewed are of Type I, as per Figure 6.5.

(5) From the input file, a method of the tube `DataModel` file is called to generate a baseline empty tube as per the initial configuration of the traffic light. This results in the creation and storage of a new `BaseTube` that acts as a placeholder for the set of durations of the

three lights. For simulations involving multiple stop lights, the same method can be called repeatedly for each set of stop lights. Each call of this method will result in a `TrafficTube` model being created and instantiated.

(6) Next, a new method is called to create and update dilemma tubes for the given input baseline tube. This leads to (a) the calling of the traffic tube instance, the extraction and storage of the set value for d_{YL}, then, (b) the creation of the dilemma tubes via an update of the traffic tube for the decreasing values of r_{YL} from d_{YL} to 0. Besides the value of r_{YL}, the values of n and n' as well as the input baseline tube are needed. The foundational variables needed to display each dilemma tube are computed, i.e., the tube type, dimensions on axis and coordinates of their location in the dimensionless (delta) space, as shown in Figure 6.7(b). The total number of dilemma tubes created is determined, as per equation (6.19). In this case, we have $n = n' = 0$, which leads to four dilemma tubes, Txx, Txo, Tox and Too which are of types I, II, III and IV, respectively.

(7) The dilemma tubes are sorted and grouped by r_{YL}. This information will allow control of the display of tubes in a way that is consistent with the unfolding of r_{YL}.

(8) With the computation and storage of dilemma tubes completed, we can now move toward their visualization. The first step consists of enabling Whistle access to the visualization tube model in order to create an instance of a JavaFX 3D chart. For those cases where the engineering simulation module is hooked to Whistle, the racetrack and its contents will be uploaded and displayed as per the set up in step (3). Otherwise, the simulation can be done with the system state in the dimensionless space computed separately based on the initial set up and targeted configurations.

(9) The 3D scene for the tube charts is created then, the data stream system is configured and the data (flow) channel tube between the input file and the 3D GUI is created and

initialized.

(10) The simulation of the engineering module is started. As the car follows the path toward the intersection stop line located at B, its position X is sensed. The remaining duration on the yellow light r_{YL} is measured from the clock. Both quantities are sent back to the computation module for processing. For each pair (XB, r_{YL}), the values of Δ_{LC}, Δ_S and Δ'_{LC} are computed as per equations (6.10), (6.11) and (6.12). As a group, these values define the state of the system in the Δ space.

(11) The set of dilemma tubes corresponding to the value of r_{YL} is pulled from storage (see step 7) and "pushed" through the channel (see step 9) to the display GUI. We can now visualize an output similar to the ones shown in Figures 6.7(c) and (d). The yellow plate is the *Plan Tube* for the system in the (Δ_{LC}, Δ'_{LC}) space. It is built from the maximum values of both parameters for the set of dilemma tubes available for display and defines the system boundary at $\Delta_S = 1$ for which the dilemma tubes take shape.

(12) Identification mechanisms are encoded into the channel system to single out *materialized tube(s)* – that is, tubes for which the safety of the system has to be checked. Materialized tubes are within the immediate vicinity of a system state and, as such, depending on how compact the tube system is, there could be many of them. There is always at least one materialized tube at any moment (in black in Figure 6.7(c) and (d)). When a materialized tube contains a system state, it means that the system is projected to be unsafe. Such tubes are qualified as "active tubes." We note that the physical interpretation of an active tube is not that of an actual violation of the system safety constraint but that it will happen in an immediate future, and certainly within the time left on the yellow light (if any).

(13) Configuration of the tube system. The way the tubes appear on the visualization

GUI depends on the values of dimensionless indexes n and n'. To identify the formation of the tubes, we look at the tubes from the top view in the plan (Δ'_{LC},Δ_{LC}) in the computer screen reference system, i.e., with Δ'_{LC} pointing downward and Δ_{LC} pointing right. As for the value of N in equation (6.19), four types of formation are possible:

$$TubeFormation = \begin{cases} point & \text{if } n \text{ and } n' \text{ are undefined} \\ line & \text{if } n \geq 0 \text{ and } n' \text{ undefined} \\ I & \text{if } n \text{ undefined and } n' \geq 0 \\ rectangle & \text{if } n' \geq 0 \text{ and } n \geq 0 \end{cases} \tag{6.20}$$

In the *point formation* the only tube that can be displayed is of Type I. In the *line formation*, realized tubes appear aligned horizontally on an axis parallel to the Δ_{LC} axis. A similar formation is observed in the *I formation* with the tubes being aligned vertically following the Δ'_{LC} in the dimensionless space. The boundary of the last type of formation has the shape of a rectangle. When $n = n'$, it becomes a square as for the four-tube formation in Figure 6.7 (c).

6.5 Safety Analyses

The purposes of this section are two-fold. First, we employ the simulation platform described in Section 6.4.2 to identify and analyze the key factors that affect the system level safety of the traffic system. In the second part of this section, single and set-pair factor safety analyses are performed to investigate how system safety depends on systematic adjustments to single factors (e.g., vehicle braking force) and combined sets of parameters.

Safety Factors Identification. Under the set of assumptions (A1) to (A3), and from Table 6.1, the following six factors are singled out for further consideration: weigh of the

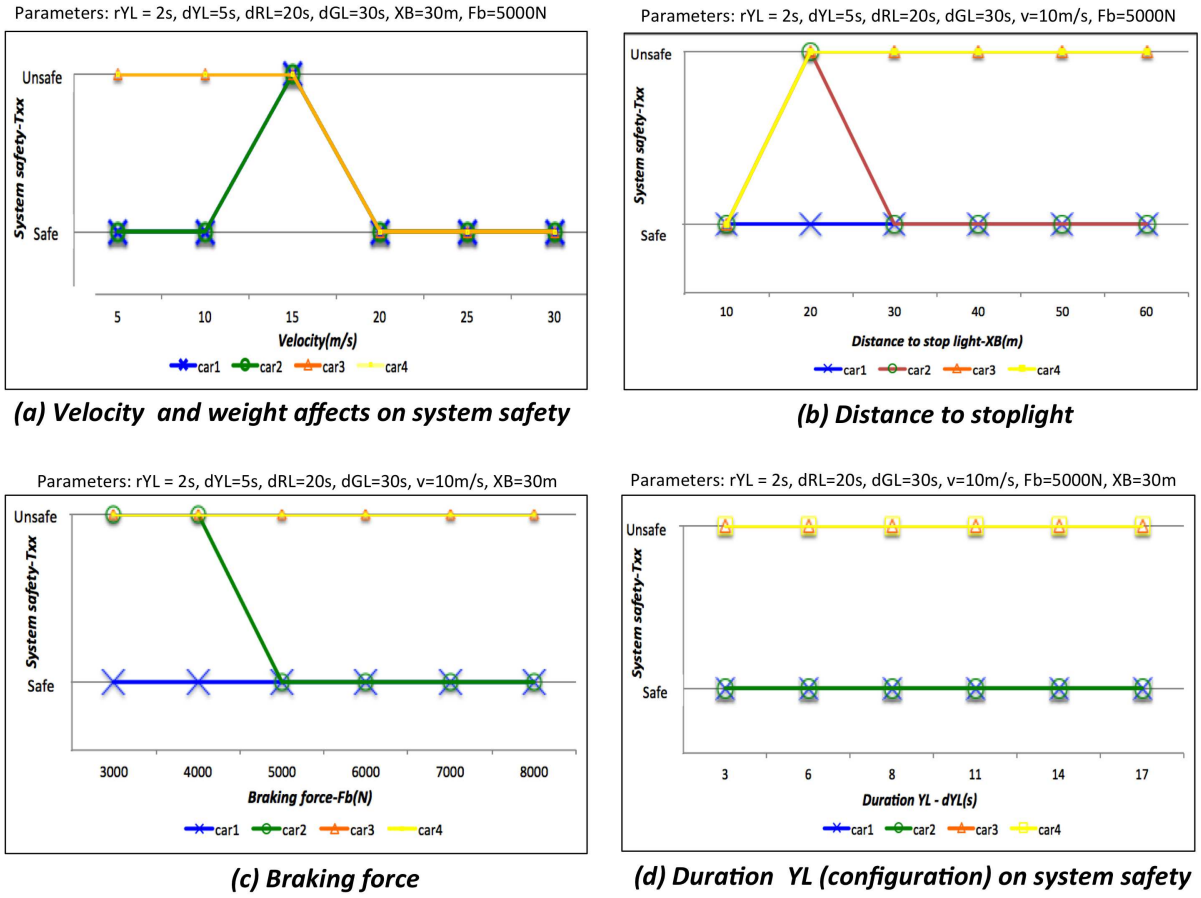


Figure 6.8: Parameters-based single factor safety profiles.

car(m), car velocity(v), car braking force(Fb), distance to stop light (XB), remaining duration of the yellow light (r_{YL}), and configured duration (d_{YL}). For these studies we pick $n = n' = 0$ which leads to a four-tube square formation.

6.5.1 Single Factor Safety Analysis

a/ Effect of Car Weight and Velocity. For this analysis, we use the set of four cars and assign for each simulation run a velocity within the range in Table 6.1 with a step of $5m/s$. The remaining four parameters are fixed to their set values. For each run, we

observe and record the presence and name of any active tube (synonym of unsafe system) as well as the identity of the car whose state has been materialized in the active tube. The absence of any active tube means the system is safe for all vehicles. The results are summarized in a *parameter-based safety profile* as shown in Figure 6.8(a).

For this particular configuration of the traffic system, the active tube for all runs is the tube Txx, which is of Type I. The heavier cars (#3 and #4) violate the safety constraint at lower speed ($v \leq 15m/s$), while small and mid-size vehicles (#1 and #2) would not violate the safety constraint if they operate on both sides of velocity $v = 15m/s$. The combined effects of inertia and velocity play against safety (i.e., heavier cars lack agility – at velocity $v \leq 15m/s$, they can neither stop before nor clear the intersection within the 2s time interval). We note the troubling "unsafe" state for all cars at $v = 15m/s$. To summarize, operating heavier vehicles within higher velocity range and, small and average size vehicle at lower or higher velocities are the only way to keep the traffic system safe.

A quick evaluation of the sensitivity of the safety profile to changes in any of the fixed parameters shows that the only one for which it doesn't change significantly is d_{YL}. For instance, if we consider changes in r_{YL}, smaller and mid-size vehicles become safer as long as r_{YL} grows beyond $2s$ ($3s$ for heavier vehicles). At lower r_{YL} ($\leq 1s$), all vehicles tend to be unsafe except for smaller ones at low velocity ($v \leq 10m/s$). Given the relatively far distance ($XB = 30$ m) at which this evaluation is performed, there might still be room for improvement as the car gets closer to the intersection stop line, especially at low velocities.

b/ Effects of the Car Distance to the Intersection. For this study, we use the same set of four cars and keep track of the distance to the stop line, this time with a step of $10m$ which is used to define the location of sensing points for the system. And as with the previous analysis, the remaining four parameters are fixed to their set value. System

safety is tracked by observing and recording the presence and name of active tubes along with the identity of the car whose state has been materialized in the active tube. Finally, the distance-to-stop-line safety profile (see Figure 6.8(b)) is generated.

We observe that, as heavier vehicles (#3 and #4) approach the intersection, they are mostly unsafe until the last checkpoint, where their dynamic capabilities allow them to either stop safely before or clear the intersection within the remaining $2s$ on the yellow light. The small vehicle (#1) is safe all the time. The mid-size vehicle (#2) is also safe at all checkpoints with the exception of checkpoint $XB = 20m$ (which corresponds to the last location where heavier vehicles transition to a safe state). An examination of the sensitivity of this profile to perturbations in r_{YL} reveals that heavier cars are more sensitive than mid-size and small cars. Away from the light ($XB \geq 50m$), heavier cars are unsafe and they will require $5s$, $4s$ and $3s$ on r_{YL},respectively at $40m$, $30m$ and $20m$ to avoid violating the intersection safety requirement. Mid-size vehicles, in contrast, only require $3s$ at $20m$ to stop.

c/ Effects of the Car Braking Force. The same protocol is followed to study how car braking force affects system safety. To that end, we systematically vary the parameter F_b within the defined range in Table 6.1 using a $1000N$ step. This results in the braking force safety profile shown in Figure 6.8(c).

For this configuration of the system, the effect of the braking force is well perceived for the mid-size car (#2) as it leaves the unsafe state when F_b increases and passes the $5,000N$ threshold. Under the same circumstances, heavier cars (#3 and #4) certainly need a braking force outside the current simulation range – in fact, our set value for the maximum force of $8,000N$ does not help switch the system back into safety. In other words, even a $8,000N$ braking force is insufficient to counter the kinetic energy of the

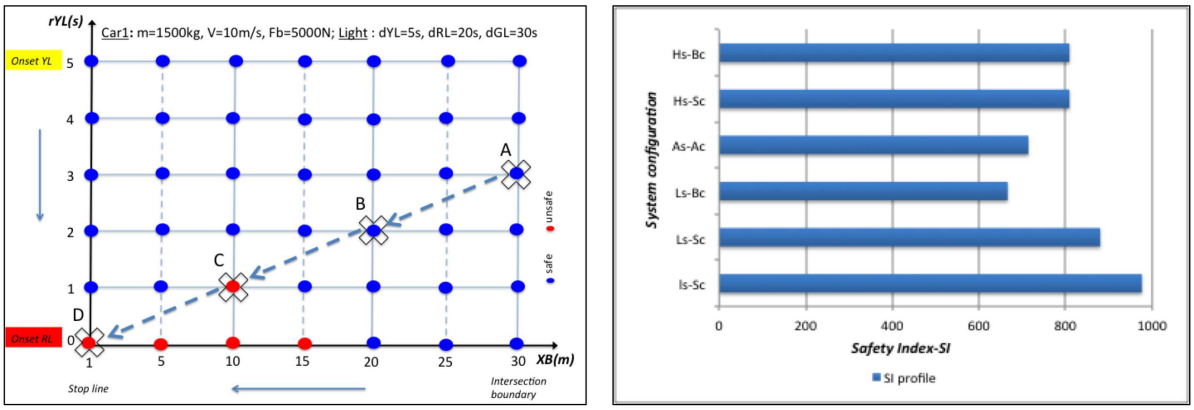
vehicles and immobilize them within $XB = 30m$ and $r_{YL} = 2s$ left on the yellow light. Small cars are much more agile, and the minimum braking force of $3,000N$ is good enough to keep the smallest car (#1) safe.

As the value of r_{YL} decreases, the safety profile for car #1 is not affected as all for all values of F_b. However, below $5,000N$, the mid-size and heavier cars would require $r_{YL} \leq 4s$ to remain safe. Above that threshold force, only heavier car will need the same amount of time to stay safe. Thus, we can conclude that the higher the inertia of the vehicle, the higher breaking force and time on yellow light are needed for the system to remain safe.

d/ Effects of the Initial Configuration of the Yellow Light. As a final step in this experiment, we would like to understand how the configuration of the stop light by the traffic engineer and, in particular, the duration of the yellow light d_{YL}, affects the system safety. To that end, we consider a fixed stop light cycle duration $C = 55s$ and assign an increasingly high percentage of that duration to the yellow light from 5% to 30% with a step of 5%; thus, the data range shown in Table 6.1. The simulation is ran for the various values of d_{YL} and results of the safety profile are shown in Figure 6.8(d).

We see from the safety profile that, for a given value of $r_{YL} = 2s$, increasing the actual duration of the yellow light does not affect the outcome of system safety. However, a look at the corresponding tube formation shows that, as the value of d_{YL} increases, so is the spacing between the tubes. This translates into more room for safety, should the system manage to get out of unsafe situations, i.e., the volume occupied by the tubes. The contrast between the tube formations in Figures 6.7(c) and (d) illustrates this phenomenon. When $d_{YL} = 5s$, a low value, the rectangle formation is compact, and the tubes are closed to each other (see Figure 6.7 (d)). Should they realize all, there will be little to no room to avoid

a violation of the safety constraint. Conversely, at higher $d_{YL} = 100s$ (for illustration only) there is plenty of room between the tubes. This means that, should there exist a mechanism to take advantage of the availability of this safety space to adjust r_{YL} to higher values, the safety of the system will be improved. *These observations make the case for reconfigurable traffic lights that are capable of adjusting the remaining duration of the yellow light to resolve safety issues.* Also, we note the variation in tube sizes in Figures 6.7(d) and 6.7(c), with Txx being the smallest and Too the biggest. This observation can be traced back to index k, as per equation (6.4), and its further propagation into the parameters that define the tubes as shown in Figure 6.5, especially those defined by equations (6.15) to (6.18). Finally, we note that $0 \leq r_{YL} \leq d_{YL}$ thus, the two variables are dependent. Setting d_{YL} from an initial position d_{YL1} to $d_{YL2} \geq d_{YL1}$ allows r_{YL} to add $d_{YL2} - d_{YL1}$ to its range which, as we have seen so far, adds more safe room for the overall system.



(a) Safety template for the set (XB,rYL) **(b) Safety Index chart for set (XB, rYL)**

Figure 6.9: Parameters-based safety templates and indexes.

6.5.2 Set (pair) Factor Safety Analysis

Despite the valuable insight provided by single factor analyses in understanding system level safety, they provide just a "snapshot" view of the system through the perspective of the parameter considered for the analysis. The sensitivity of most safety profiles to changes in the values of r_{YL} clearly shows that even though most factors are set or controlled independently, their interaction is the key driver behind system level safety. Thus, there is a need to look at changes to system safety caused by adjustments to combined sets of parameters.

a/ Parameter-based Safety Template for Pair (r_{YL}, XB). Pairing the six parameters leads to fifteen possible sets. However, given that parameters such as r_{YL} and d_{YL} are dependent and others such as m and XB are constrained by the vehicle physics, not two sets of parameters are equally important or relevant for this study. Thus, we won't be analyzing the system safety for all pairs, but we will be looking at the pair (r_{YL}, XB), which illustrates the cyber-physicality of the traffic system as introduced in Section 6.3.1. The protocol of the study described here can be repeated and applied to other pairs as well.

For set factor studies, all the parameters considered vary within their individual, predefined range. The other parameters are configured to their set values as presented in Table 6.1. Running the simulation and recording the safety state of the system result in the creation of a parameter-based *safety template*, such as the one seen on Figure 6.9(a). This particular template is created with the configuration $K \equiv (m = 1,500kg, v = 10m/s, Fb = 5,000N, d_{YL} = 5s, d_{RL} = 20s, d_{GL} = 30s)$. The template shows the safety

state of each system operational point. A red dot signifies that under K, the system state is in an active tube (i.e., the system is unsafe). A blue dot means the system is safe. In practical terms, the template is an indicator of safety – for instance, under configuration K, if car #1 crosses the intersection boundary (XB=30m) when there is only $3s$ left on the yellow light, the system will be safe as it will be located at $A(30m, 3s)$, which is a safe operational point on the template. If, however, the configuration K remains unchanged, the system will be unsafe $2s$ later at location $C(10m, 1s)$. Therefore, for the system to remain safe under K, the car has to enter the intersection when there is at least $4s$ left on the yellow light. These examples illustrate the greater insight, we can gain using safety templates, in the interplay between system parameters and their effects on system level safety.

b/ Parameter-based Safety Indexes for Pair (r_{YL}, XB). A subspace Us that contains all unsafe states of the system for the configuration K can be defined as follows.

$$Us^{K}_{(r_{YL},XB)} = \begin{cases} 0s \leq r_{YL} \leq 1s \\ \\ 1m \leq XB \leq 15m. \end{cases} \tag{6.21}$$

Intuitively, one might think that a smaller subspace Us translates to a safer system, but this is only part of the story. Considering that an unsafe subspace might also contain safe states, as observed in this case, we ought to be able to quantitatively assess the safety of a configuration in a clear and simple way. To this end, we introduce the parameter-based *configuration safety index* SI as follows.

$$SI^{K}_{(r_{YL},XB)} = \left(1 - \frac{n_{U_K}}{n_K}\right) * 1000. \tag{6.22}$$

Here, n_{U_K} is the number of unsafe states (red dots) in Us and n_K the total number of states in the template for configuration K. For the safety template shown in Figure 6.9(a), we count $n_{U_K} = 5$ unsafe states and $n_K = 6 * 7 = 42$ total states. This leads to a configuration safety index of $SI^K_{(r_{YL},XB)} = 880$.

By systematically adjusting the vehicle weight (m) and velocity (v) we can generate an ensemble of safety templates, and then for each, compute the safety index. This leads to the safety index chart shown in Figure 6.9(b). The chart shows that for high speeds, both the smallest vehicle (S_c) and heaviest vehicle (B_c) have similar levels of safety. The smallest vehicle does a better job at lower velocities. In-between, the mid-size vehicle (A_c) cannot do better at average velocity (A_s).These results are consistent with the findings in Section a/.

We note that this safety index does not capture the topology of unsafe and safe points in the Us subspace for (r_{YL}, XB). As seen in Section a/ above, that distribution is critical in predicting the future state of the system. Therefore, we cannot use the safety index SI to that same end. However, it can be used for a high level estimate of the parameter-based safety appreciation of the system safety before diving into topological considerations of Us for further investigation. To that extent, the two approaches serve complementary purposes.

6.5.3 Beyond Predefined Configurations and Pair Factors

Any change in the value of a parameter in the configuration K in Section 6.5.2 a/ automatically forces the switch to a different safety template (with the new value for that parameter) to predict the state of the system when the car reaches the stop line. This limits the ability of the Systems Engineer to navigate the design space of the

traffic system. A possible solution is to flatten all independent variables in a pentagon-like diagram which will give a partial view of the whole design space. The actual full design space is much more complex (i.e., a five-dimensional shape) and almost impossible to visualize. Any combination of values of the five parameters (m, v, F_b, r_{YL}, XB), each within its respective range, is theoretically a valid point.

6.6 Discussion

Our preliminary results are contingent upon assumptions (A1) through (A3) listed in Section 6.4.2. Neglecting air resistance (A1) certainly simplifies the account of the dynamics of the cars but it comes at a price. With the acceleration null, the velocity is assumed constant on XB which leads to a constant value of Θ_B in equation (6.10) for all vehicles at the same velocity for the same value of XB. This propagates all the way to the tubes visualization where, under such circumstances, points for the various cars will be stuck in the plan (Δ_S, Δ'_{LC}) at a single Δ_{LC} value. One opportunity for further investigation is to account for the air resistance in the dynamics of the car, through a drag force $f = k_1 * v^2$ for instance. This will lead to a more accurate model of the vehicle dynamic that will ultimately improve the quality of the results. The immediate effect on the tube framework will be the distribution of system states along the axis Δ_{LC} as well.

Task execution of the scenario introduced in Section 6.4.2 requires intensive computations and communication at multiple steps; this makes it hard for assumptions (A2) and (A3) to survive any physical prototype testing of the system. In fact, as many researchers have pointed out, not only do real-world computations and communication require finite amounts of time to complete [156, 283], but delays of unacceptable duration

can trigger accidents in traffic scenarios that are safety critical. Given that such considerations are platform-dependent, there should be in a future iteration of this work a mechanism to account for delay information in the execution model, perhaps along the lines of what has been accomplished with Ptolemy [221].

Chapter 7: Metric and Spatio-Temporal Algorithms for Safety-Critical Cyber-Physical Systems

7.1 Introduction

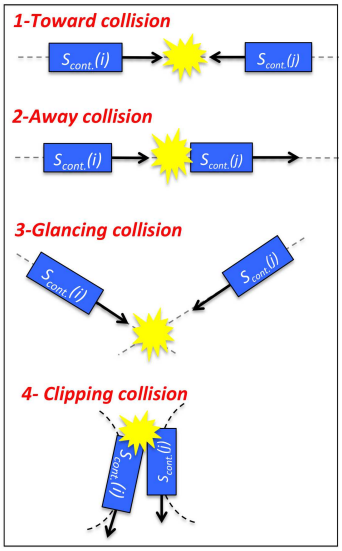


Figure 7.1: Types of collision.

Whether desired or not, collisions among objects happen as a result of (unresolved) spatio-temporal conflicts between the entities involved. Preventing such accident to occur in safety-critical CPS requires not just the correct predictions of future time-based system state as seen in Chapter 4, but also its spatial-based state. In other words, it's critical that the system makes the right decision and takes the right action at the right time and right place to remain safe. Therefore, we ought to investigate and understand spatial semantics as well as the interplay between time and space theories in supporting successful spatio-temporal conflict resolution algorithms and strategies. Thus, in this chapter we will examine different types of collisions then, develop metrics for characterizing safety in this context and finally, construct algorithms to prevent the occurrence of such collisions.

In Section 7.2, we investigate the classification of collisions that will be used subsequently in this chapter. In Section 7.3, spatial models introduced in Chapter 4 are used to conceptualize and develop safety tubes and metrics for dynamic entities in away collision. Section 7.4 brings together the power of meta-domain (i.e., space and time) semantics and metrics to develop safety algorithms for away, glancing and clipping collisions avoidance. Finally, Section 7.5 demonstrates the effectiveness of the spatio-temporal based approach in tackling the problem of glancing collision at a non-signalized traffic intersection. We show that the ontological commitment (i.e., how the world is seen) of object representations with regard to the dimension of the space is critical to the proper understanding of the spatial configuration of the system in the world by the cyber (on-board computer) and the accurate prediction of the collision.

7.2 Types of Collision

In spatially distributed systems, safety challenges are often materialized in the form of risks of collision. Collision between dynamic entities is a permanent concern and has led researchers to investigate and develop strategies, algorithms and systems to avoid collision in many industries including aerospace [122], automobile [168] and railway [266]. Generally speaking, and as illustrated in Figure 7.1, scholars [80] have categorized collisions in four types:

(1) *Toward collision.* Also called *face-to-face* collision, it occurs when the entities collide while moving toward each other while on the same trajectory.

(2) *Away collision.* In this type of collision, the two entities are moving on the same trajectory and the one behind rears end the one ahead (or an obstacle).

(3) *Glancing collision.* Entities in this type of collision are moving on different but crossing trajectories and the agents collide at the intersection point of the two trajectories.

(4) *Clipping collision.* The two entities involved in this collision are traveling on non-parallel yet, non-intersecting trajectories but they end up colliding because of their shape. Examples include taxiway incidents at crowded airports around the globe [14, 138]. The study of this type of collision requires spatial models of dimension D ≥ 2.

7.3 Tubes and Metrics for Dynamic Entities on Away Collision Course

In this section, we use the prospective on spatial modeling introduced in Section 4.4 to revisit the lane control problem and develop safety metrics and tubes in that context. The lane control problem is cast as an away collision challenge as defined in Section 7.2.

7.3.1 Objectives and Modeling Assumptions

Background and Objectives. In [170], the authors introduce and verify formal models for distributed cooperative control of multiple cars. They form distributed hybrid systems where components coordinate their actions in order to minimize the risk of collisions. The resulting systems are cyber-physical systems where cyber functionality (V2V and V2I communication, computation, control) have to closely interact and coordinate their action with physical functionality (sensing, actuation) at various level of complexity to keep the system safe locally and globally. The models are formulated as quantified hybrid programs (QHP) that account for both the dynamic and the control of individual cars and the way

they affect system safety. This model is verified using a formal proof calculus for QdL [216]. Space is captured in those models at a 0D level which, in light with issues raised by certain types of collisions e.g. away & clipping collisions (see Section 7.2), is not appropriate for a complete understanding and resolution of the problem. Thus, we seek to generalize, extend and use those models in the context of the various representations of spatial entities as introduced in Section 4.4. Our main focus will be the preservation of the various results obtained, in the face of increasingly explicit and expressive spatial representations of the dynamic entities. Also, we seek to develop and define safety metrics for each appropriate level of refinement of spatial representations for a quick, easy evaluation of the system safety.

Assumptions. We reiterate here some of the key assumptions made for this work.

(A1) The acceleration a of each dynamic entity takes instant effect and its global maximum limit is denoted A.

(A2) The braking power of each dynamic entity varies between b (minimum) and B (maximum) with $B > b > 0$.

(A3) The reaction time for each entity is bounded by ε which can physically be tied to the inverse of the frequency at which sensors are updated in the system.

(A4) Each entity, except for the first one in the lane, will have at most one leader and they are all moving forward only.

(A5) Each system component (e.g. vehicle, aircraft, etc.) is a rigid uniform, non-deformable body.

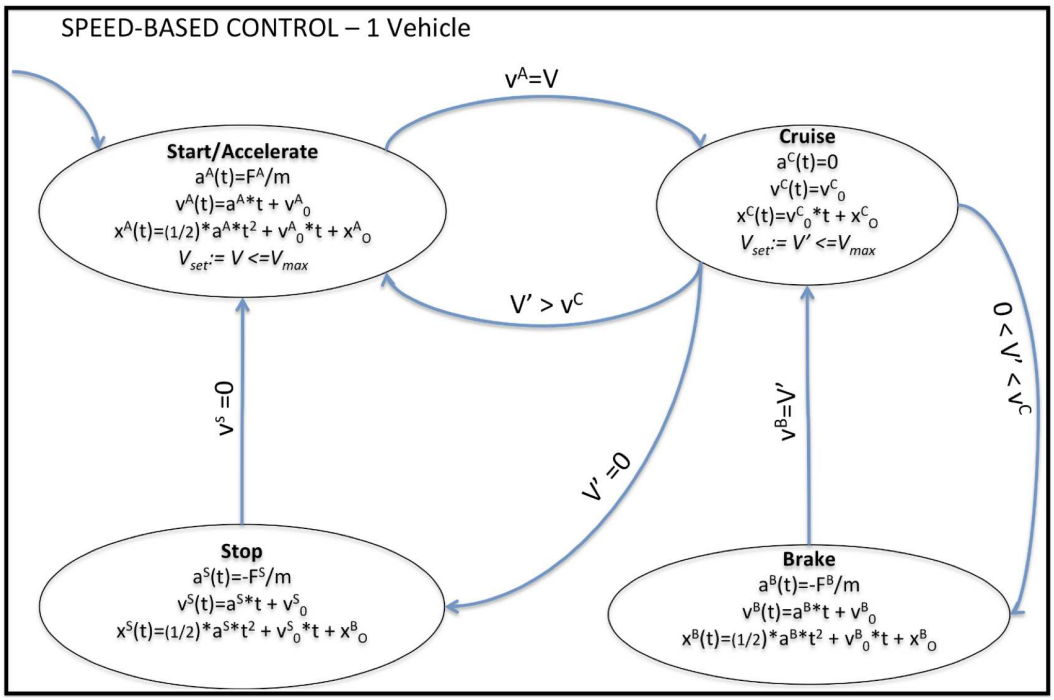


Figure 7.2: Behavior of a single vehicle as a Hybrid system.

7.3.2 Local and Global Lane Safety

The behavior of a controlled dynamic entity (taken alone) such as a vehicle, can be modeled as a hybrid system as shown in Figure 7.2. From a stopping state, the vehicle can accelerate to a set speed V_{set} and, once it's reached, will cruise until another one is set. If the new set speed is smaller than the previous then the vehicle will slow down (brake) or stop if the new speed is null. Otherwise, the vehicle will go back to the acceleration state and increase its speed.

Local lane control and safety. In a situation where the vehicle is part of a 2-vehicle system on a longitudinal lane, this simple and straightforward mode of operation may not be possible especially if the given vehicle is the follower (f) of a lead (l) one. As shown in [170], the lead vehicle can still behave as described on Figure 7.2 but the follower can only do the same when it's safe; in other words, when the following safety condition is

satisfied.

$$Safe_\varepsilon \equiv x_f + \frac{v_f^2}{2b} + \left(\frac{A}{b} + 1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon v_f\right) < x_l + \frac{v_l^2}{2B} \tag{7.1}$$

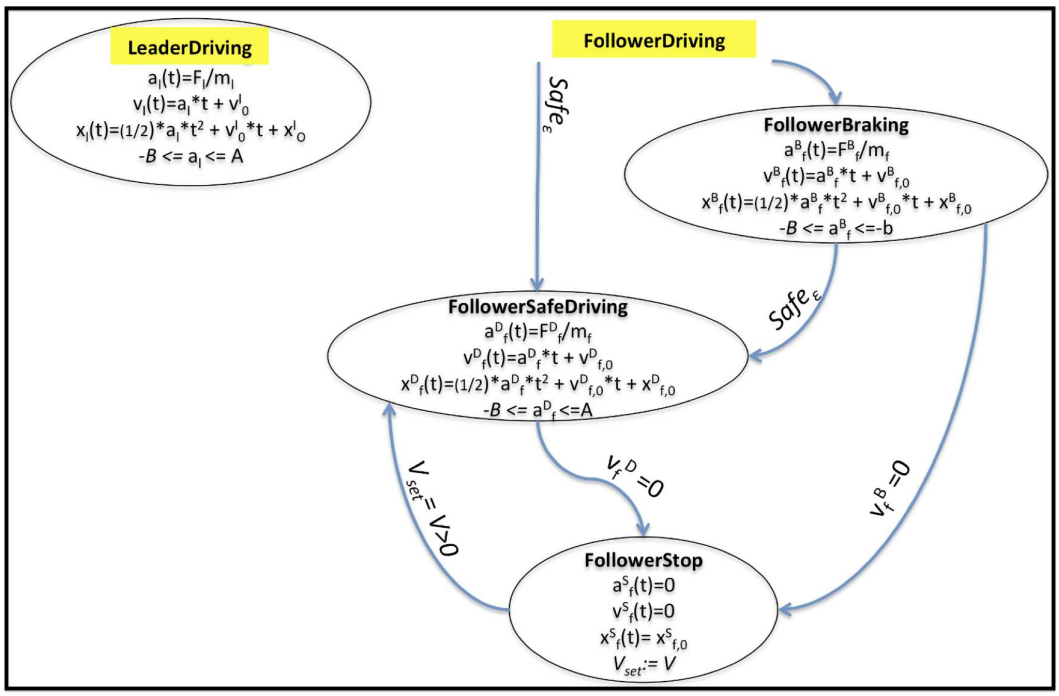


Figure 7.3: Behavior of leader and follower vehicles as Hybrid systems.

The follower l can brake anytime on it own without having to acknowledge the decision made by f. However, this will not always prevent a collision. This can be explained by the fact that if it's so close to l that even the hardest instant braking would not prevent a crash, the system is already unsafe and the decision to brake would ultimately not matter. Figure 7.3 illustrates the resulting behavior of the follower vehicle. For the purpose of simplification the behavior of the leader has been kept simple. For the 2-vehicle system to remain safe locally i.e. avoid away collision situation, f has to remain behind l all the time. Thus, the following *Safe distance formula* relation must be a system invariant

that must hold.

$$(f \ll l) \rightarrow \left(x_f < x_l \wedge x_f + \frac{v_f^2}{2b} < x_l + \frac{v_l^2}{2B} \wedge v_f \geq 0 \wedge v_l \geq 0 \right) \tag{7.2}$$

Global lane control and safety. In the context of global lane control challenge, the system is made of $n > 2$ vehicles moving forward on a longitudinal lane, with none allowed to pass the one in front. Using previous results, first-order variables and assumption (A4), the leader of a given vehicle i in the list of vehicle C is defined as follows.

$$L(i) = j \equiv x(i) < x(j) \wedge \forall k : C \setminus \{i, j\} \, (x(k) < x(i) \vee x(j) < x(k)) \tag{7.3}$$

$$(i \ll L(i)) \equiv \forall j : C \, ((L(i) = j) \rightarrow (i \ll j)) \tag{7.4}$$

The verification of the safety of the global level system safety involves the definition of transitive leaders $L^*(i)$ as follows.

$$(i \ll L^*(i)) \equiv [k ::= i; (k ::= L(k))^*](i \ll k) \tag{7.5}$$

Ultimately, it's shown that for every configuration of the system in which each car is safely following the car directly in front of it, all cars will remain in a safe configuration while they follow the distributed control. Thus, the following *Safety formula* must hold all the time for the system to be safe.

$$\forall i : C \, (i \ll L(i)) \rightarrow [glc](\forall i : C \, (i \ll L^*(i))) \tag{7.6}$$

Additional information, details and proofs can be found in [170].

Limitations of Lane Control and Safety Models. The models and corresponding safety invariants introduced in this Section are based on a space-point theorization of the space occupied by each vehicle. Thus, that space is captured by the 0D representations of vehicles as points i.e. $x_l, x(i)$. Thus, the shape, length and actual boundaries of the vehicles are ignored in the model. As shown in Section 4.4, the granularity and fidelity of the representation of spatial entities are critical in safety-related decisions. Moreover, the models explicitly reduce the scope of the results to configurations of the system for which the entities move along longitudinal lanes. In real world, they have to negotiate curves and uneven terrains. Therefore, we need to revisit system safety invariant formulas with respect to the various levels of representation of spatial entities (as pictured in Figure 4.3).

7.3.3 Local Lanes Safety Formulas for Away Collision

Space-based Safety Invariant Formulas. From assumption (A5), all points within the boundary of the spatial entity representing a moving component (vehicle) will move at the same speed. Therefore, it guarantees that the local and global lane control models introduced in [170] remain valid if the granularity of the spatial entities representation changes (increases). However, there is a need to clearly specify in the models which point(s) of the entity(ies) is(are) been used by the models. This is particularly critical for the safety invariants formulas as they have to properly capture and express the conditions under which the system is safe. Consequently, we look into the (re)formulation of the safety invariant relation (7.2) for local lane control for each of the 4 dimensions (0,1,2,3) of space as defined in Section 4.4.3. In order to account to real-world situations, the formulation of the invariants should consider curved trajectories in general while remaining true to longitudinal (straight lines) ones too. Accounting for the curvature of the trajectory

constraints pushes us to consider the orientation of the object/entity as well as its velocity and acceleration - with respect to a given reference system - and the need to integrate them in the formulation of the right hand side of equation (7.2).

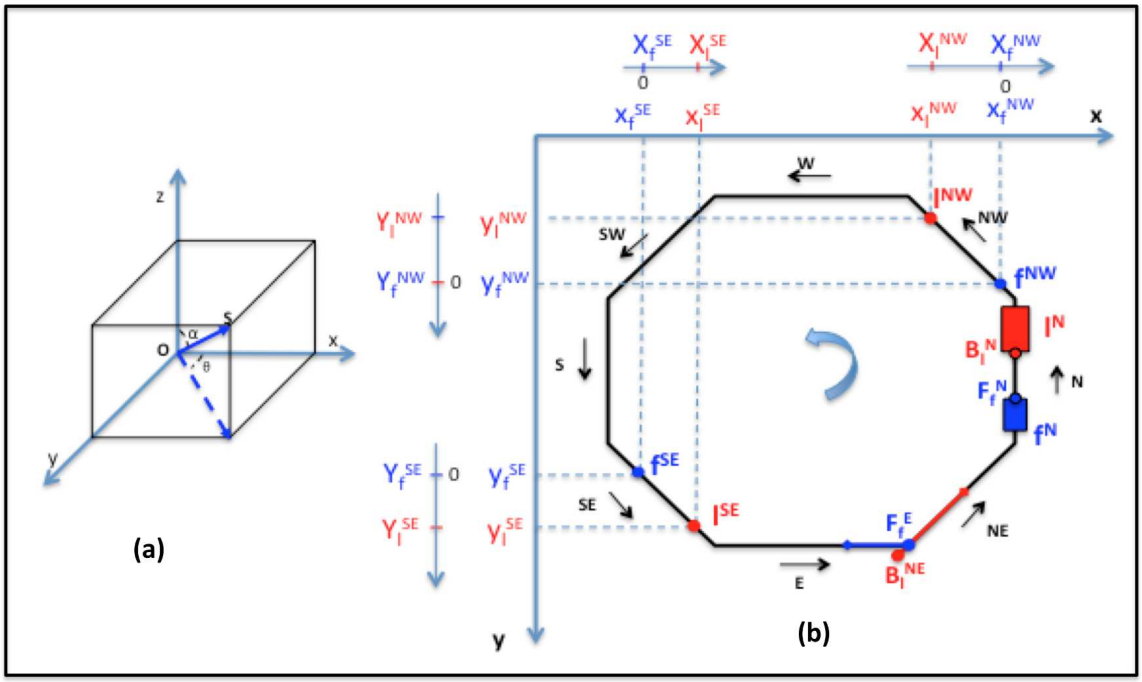


Figure 7.4: 3D coordinates representation (a) and illustrative cardinal orientation track in 2D space (b).

1/0D space model. Under this perspective, objects are represented by their center of inertia (as a point), where the displacement force is applied. In a general case, we represent the object as a point in 3D coordinate system as shown for the point S in Figure 7.4 (a). In order to enable the representation of the system on longitudinal axes, we define a transformation (T) as follows.

$$(T) \begin{cases} X = x - x_f \\ Y = y - y_f \\ Z = z - z_f \end{cases} \tag{7.7}$$

where f is the following entity or the "follower".

Using Figure 7.4 (a), the orientation vector **ov** of the object is defined as follows.

$$\mathbf{ov}^{(R)} \begin{cases} ov^X = sin(\alpha)cos(\theta) \\ ov^Y = sin(\alpha)sin(\theta) \\ ov^Z = cos(\alpha) \end{cases} \tag{7.8}$$

with $\|\mathbf{ov}\| = 1$, is the norm of the orientation vector **ov**.

The representative coordinates of the leader, follower as well as their respective velocity and acceleration vectors are determined in R=(X,Y,Z) coordinate system as follows.

Position leader:

$$\mathbf{l}^{(R)} \begin{cases} X_l = x_l - x_f \\ Y_l = y_l - y_f \\ Z_l = z_l - z_f \end{cases} \tag{7.9}$$

Position follower:

$$\mathbf{f}^{(R)} \begin{cases} X_f = 0 \\ Y_f = 0 \\ Z_f = 0 \end{cases} \tag{7.10}$$

Velocity leader:

$$\mathbf{v}_l^{(R)} \begin{cases} v_l^X = V_l ov_l^X = V_l sin(\alpha_l)cos(\theta_l) \\ v_l^Y = V_l ov_l^Y = V_l sin(\alpha_l)sin(\theta_l) \\ v_l^Z = V_l ov_l^Z = V_l cos(\alpha_l) \end{cases} \tag{7.11}$$

where V_l is the norm of the velocity vector $\mathbf{v}_l$. Similarly, the velocity of the follower is determined as follows.

Velocity follower:

$$\mathbf{v}_f^{(R)} \begin{cases} v_f^X = V_f ov_f^X = V_f sin(\alpha_f) cos(\theta_f) \\ v_f^Y = V_f ov_f^Y = V_f sin(\alpha_f) sin(\theta_f) \\ v_f^Z = V_f ov_f^Z = V_f cos(\alpha_f) \end{cases} \tag{7.12}$$

where V_f is the norm of the velocity vector $\mathbf{v}_f$.

The acceleration is defined as the derivative of the velocity i.e. $\mathbf{v}' = \mathbf{a}$.

Acceleration leader:

$$\mathbf{a}_l^{(R)} \begin{cases} a_l^X = v_l'^X ov_l^X + v_l^X ov_l'^X \\ a_l^Y = v_l'^Y ov_l^Y + v_l^Y ov_l'^Y \\ a_l^Z = v_l'^Z ov_l^Z + v_l^Z ov_l'^Z \end{cases} \tag{7.13}$$

After calculation, we obtain:

$$\mathbf{a}_l^{(R)} \begin{cases} a_l^X = V_l \alpha_l' cos(\alpha_l) cos(\theta_l) - V_l \theta_l' sin(\alpha_l) sin(\theta_l) \\ a_l^Y = V_l \alpha_l' cos(\alpha_l) sin(\theta_l) + V_l \theta_l' sin(\alpha_l) cos(\theta_l) \\ a_l^Z = -V_l \alpha_l' sin(\alpha_l) \end{cases} \tag{7.14}$$

Acceleration follower:

$$\mathbf{a}_f^{(R)} \begin{cases} a_f^X = v_f'^X ov_f^X + v_f^X ov_f'^X \\ a_f^Y = v_f'^Y ov_f^Y + v_f^Y ov_f'^Y \\ a_f^Z = v_f'^Z ov_f^Z + v_f^Z ov_f'^Z \end{cases} \tag{7.15}$$

As for the leader, we obtain after calculation:

$$\mathbf{a}_f^{(R)} \begin{cases} a_f^X = V_f \alpha_f' cos(\alpha_f) cos(\theta_f) - V_f \theta_f' sin(\alpha_f) sin(\theta_f) \\ a_f^Y = V_f \alpha_f' cos(\alpha_f) sin(\theta_f) + V_f \theta_f' sin(\alpha_f) cos(\theta_f) \\ a_f^Z = -V_f \alpha_f' sin(\alpha_f) \end{cases} \tag{7.16}$$

As per assumptions (A1) and (A2), we have $-B \leq a_k^i \leq A$ with $i \in \{X, Y, Z\}$ and $k \in \{l, f\}$.

Figure 7.4 (b) illustrates the representation of moving vehicles - red dot for the leader and blue dot for the follower - on an octagon track. The eight branches of the octagon capture the main orientations of the entity's trajectory in 2D space. The workspace is augmented with the coordinates of the objects when projected on the (X,Y) axes of the (R) coordinate system. These axes are in the same orientations and are parallel to their counter part x and y. We note that the direction in which the track is traveled and the branch on which the movement is tracked affect the sign of the coordinate of the leader. However, the coordinate of the follower is always null (origin). In (R) coordinate system, each axis is an oriented longitudinal coordinate system as considered above. Thus, we seek to write the right hand side of equation (7.2) for each of the 3 axes. To that aim, we introduce an orientation vector **oa** for the acceleration of the object as follows.

$$\mathbf{oa}^{(R)} \begin{cases} oa^X = sin(\beta)cos(\lambda) \\ oa^Y = sin(\beta)sin(\lambda) \\ oa^Z = cos(\beta) \end{cases} \tag{7.17}$$

where $\|\mathbf{oa}\| = 1$ is the norm of the orientation vector **oa**. Angles β and λ are the counterpart of α and θ introduced in equation 7.8. We rewrite the acceleration of the vehicle as follows.

$$\mathbf{a}^{(R)} \begin{cases} a^X = A * oa^X = Asin(\beta)cos(\lambda) \\ a^Y = A * oa^Y = Asin(\beta)sin(\lambda) \\ a^Z = A * oa^Z = Acos(\beta) \end{cases} \tag{7.18}$$

where A is the norm of the acceleration vector **a**. The values of β, λ and A can be calculated by putting side by side any of equations (7.14) or (7.16) and (7.18). This leads

to the following: $\lambda = \arctan(v'_y/v'_x)$, $\beta = \arccos(v'_z/A)$ with $A = (v'^2_x + v'^2_y + v'^2_z)^{1/2}$

The minimum and maximum braking distances (in the worse case scenarios) in the second part of each expression of the invariant are obtained as follows. We first notice that during braking, for $i \in \{X, Y, Z\}$, $a^i < 0$. Then, from assumption (A2) on breaking powers, we have $b < \|\mathbf{a}\| < B$. We then use the formulation in equation (7.18) to represent the acceleration. Therefore, the braking distance bd on each axis for the follower is bounded by the projection of its maximum ($\|\mathbf{a}\| = b$) value as follows.

$$\mathbf{bd}_f^{(R)} \begin{cases} bd_f^X = \frac{(v_f^X)^2}{2|a_f^X|} < \frac{V_f^2 sin^2(\alpha_f) cos^2(\theta_f)}{2b|sin(\beta_f) cos(\lambda_f)|} \\ bd_f^Y = \frac{(v_f^Y)^2}{2|a_f^Y|} < \frac{V_f^2 sin^2(\alpha_f) sin^2(\theta_f)}{2b|sin(\beta_f) sin(\lambda_f)|} \\ bd_f^Z = \frac{(v_f^z)^2}{2|a_f^Z|} < \frac{V_f^2 cos^2(\alpha_f)}{2b|cos(\beta_f)|} \end{cases} \tag{7.19}$$

with $\alpha_f, \theta_f, \beta_f, \lambda_f \notin \{k\pi/2, (2k+1)\pi/2\}$ and $k \in \mathbb{Z}$.

Similarly, the braking distance for the leader is bounded by its minimum value as follows.

$$\mathbf{bd}_l^{(R)} \begin{cases} bd_l^X = \frac{(v_l^X)^2}{2|a_l^X|} > \frac{V_l^2 sin^2(\alpha_l) cos^2(\theta_l)}{2B|sin(\beta_l) cos(\lambda_l)|} \\ bd_l^Y = \frac{(v_l^Y)^2}{2|a_l^Y|} > \frac{V_l^2 sin^2(\alpha_l) sin^2(\theta_l)}{2B|sin(\beta_l) sin(\lambda_l)|} \\ bd_l^Z = \frac{(v_l^z)^2}{2|a_l^Z|} > \frac{V_l^2 cos^2(\alpha_l)}{2B|cos(\beta_l)|} \end{cases} \tag{7.20}$$

with $\beta_l, \lambda_l \notin \{k\pi/2, (2k+1)\pi/2\}$ and $k \in \mathbb{Z}$.

Therefore, the bound values in equations (7.19) and (7.20) illustrate the worse case scenarios for which the leader applies the maximum braking power B while the follower applies the minimum value b. Table 7.1 shows the resulting expression of the safety invariant function obtained for each axis.

The following special cases are of interest.

Case 1.1: $\alpha_f = \alpha_l = k\pi, k \in \mathbb{Z}$. This implies $sin(\alpha_f) = sin(\alpha_l) = 0$ and $cos(\alpha_f) =$

$$|X_f| < |X_l| \wedge |X_f| + \frac{V_f^2 sin^2(\alpha_f) cos^2(\theta_f)}{2b|sin(\beta_f) cos(\lambda_f)|} < |X_l| + \frac{V_l^2 sin^2(\alpha_l) cos^2(\theta_l)}{2B|sin(\beta_l) cos(\lambda_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \quad (7.21)$$

$$|Y_f| < |Y_l| \wedge |Y_f| + \frac{V_f^2 sin^2(\alpha_f) sin^2(\theta_f)}{2b|sin(\beta_f) sin(\lambda_f)|} < |Y_l| + \frac{V_l^2 sin^2(\alpha_l) sin^2(\theta_l)}{2B|sin(\beta_l) sin(\lambda_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \quad (7.22)$$

$$|Z_f| < |Z_l| \wedge |Z_f| + \frac{V_f^2 cos^2(\alpha_f)}{2b|cos(\beta_f)|} < |Z_l| + \frac{V_l^2 cos^2(\alpha_l)}{2B|cos(\beta_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \quad (7.23)$$

Table 7.1: Safety invariant formulas for 0D space model

± 1; $cos(\alpha_l) = \pm 1$. Plus, the acceleration and velocity vectors are parallel i.e. $\beta_f, \beta_l = k\pi, k \in \mathbb{Z}$. This leads to $cos(\beta_f) = \pm 1$ and $cos(\beta_l) = \pm 1$. Thus, both vehicles move on the Z axis and we obtain in equation (7.23) a relation similar to the one in equation (7.2).

Case 1.2: $\alpha_f = \alpha_l = (2k+1)\pi/2$ *and* $\theta_f, \theta_l = k\pi$, *with* $k \in \mathbb{Z}$. Both vehicles move on the X axis. As above, the acceleration and velocity vectors are parallel i.e. $\beta_f, \beta_l = (2k+1)\pi/2$ and $\lambda_f, \lambda_l = k\pi$, with $k \in \mathbb{Z}$. The exact same relation expressed by equation (7.2) is found for equation (7.21).

Case 1.3: $\alpha_f = \alpha_l = \pi/2$; $\theta_f = (2k+1)\pi/2$; $\theta_l \neq (2k+1)\pi/2$; $k \in \mathbb{Z}$ It results a configuration of the system for which follower and leader are on different, non-parallel segments on a (X,Y) plan such as the one on Figure 7.4. One such configuration could be when leader and follower are respectively on the north (N) and north west (NW) or west (W) and south east (SE) branches of the track.

From this 0D space model and, assuming a representation of the objects in 2D or 3D, we will need to separate the leader and the follower in ONLY ONE (1) of the three dimensions in order to keep the system safe. Thus, only one of the conditions (7.21), (7.22), (7.23) needs to be satisfied for the system to be safe.

2/1D space model. With 1D space models, each object is represented by two

points materializing their front (F) and back (B) as pictured in center of Figure 4.3. The 2 points are actually the extremities of the line segment that represents the object. Under assumption (A5) both front and back points - as part of a rigid body - have the same dynamic, thus they share the same acceleration and speed but not the same position in space. For the system to remain safe, we need to guarantee that the back of the leader (l) is not going to be hit by the front of the follower (f). Therefore, the two points we should be tracking are their respective back and front i.e. F_f and B_l. Thus, we'll consider the position, velocity, acceleration and orientation of those particular points of the 2 objects and rewrite the safety invariant conditions found in Table 7.1 as shown in Table 7.2.

$$|X_f^F| < |X_l^B| \wedge |X_f^F| + \frac{V_f^2 sin^2(\alpha_f^F) cos^2(\theta_f^F)}{2b|sin(\beta_f^F) cos(\lambda_f^F)|} < |X_l^B| + \frac{V_l^2 sin^2(\alpha_l^B) cos^2(\theta_l^B)}{2B|sin(\beta_l^B) cos(\lambda_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.24}$$

$$|Y_f^F| < |Y_l^B| \wedge |Y_f^F| + \frac{V_f^2 sin^2(\alpha_f^F) sin^2(\theta_f^F)}{2b|sin(\beta_f^F) sin(\lambda_f^F)|} < |Y_l^B| + \frac{V_l^2 sin^2(\alpha_l^B) sin^2(\theta_l^B)}{2B|sin(\beta_l^B) sin(\lambda_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.25}$$

$$|Z_f^F| < |Z_l^B| \wedge |Z_f^F| + \frac{V_f^2 cos^2(\alpha_f^F)}{2b|cos(\beta_f^F)|} < |Z_l^B| + \frac{V_l^2 cos^2(\alpha_l^B)}{2B|cos(\beta_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.26}$$

Table 7.2: Safety invariant formulas for 1D space model

Remark 7.1 *(Some limitations and a solution).* The accuracy and precision of the safety model are critical for the effectiveness of the reasoning process. Thus, the effectiveness of this safety invariant formulas is based on the premises that (1) each front and back point of the objects follow a predefined trajectory without any slip and (2) the design of the trajectories is appropriate with the physics of the vehicle. In real world applications, situations such as bad weather (snow, heavy rain, ice), length of the vehicle with regard to the acuity of curve and turns can result to loss of control of either the front or back of the vehicle or both. Therefore, a system configuration such as the one at the bottom right of

Figure 7.4 b) is possible. This situation is similar to *case#1.3* above with $\alpha_f = \alpha_l = \pi/2$; $k = 0$; $\theta_l \neq 0$. On top of that, in case the leader has stopped i.e. $V_l = 0$, the invariant formula (7.25) will be satisfied all the time meaning the system is safe, which is not obviously the case as shown in Figure 7.4! Therefore, we need to account for the full length of the line segment representing the vehicle - as a rigid body - in the formula.

Resolving this issue necessitates the representation of each vehicle as 1D proper spatial (RCC-compliant) entity as introduced in the table in Figure D.3 in Appendix D. The (spatial) interaction between the leader and the follower will therefore be defined by the function f_{crash} as formulated in equation (D.9). Thus, we replace the distance constraint in the safety invariant formulas stated in Table 7.2 and obtain a revised formulation of the safety invariant property as shown in Table 7.3. Using this new formulation, it

$$(f^p_{crash}(l,f) = 0)\,|_{1D} \wedge |X_f^F| + \frac{V_f^2 sin^2(\alpha_f^F) cos^2(\theta_f^F)}{2b|sin(\beta_f^F) cos(\lambda_f^F)|} < |X_l^B| + \frac{V_l^2 sin^2(\alpha_l^B) cos^2(\theta_l^B)}{2B|sin(\beta_l^B) cos(\lambda_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.27}$$

$$(f^p_{crash}(l,f) = 0)\,|_{1D} \wedge |Y_f^F| + \frac{V_f^2 sin^2(\alpha_f^F) sin^2(\theta_f^F)}{2b|sin(\beta_f^F) sin(\lambda_f^F)|} < |Y_l^B| + \frac{V_l^2 sin^2(\alpha_l^B) sin^2(\theta_l^B)}{2B|sin(\beta_l^B) sin(\lambda_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.28}$$

$$(f^p_{crash}(l,f) = 0)\,|_{1D} \wedge |Z_f^F| + \frac{V_f^2 cos^2(\alpha_f^F)}{2b|cos(\beta_f^F)|} < |Z_l^B| + \frac{V_l^2 cos^2(\alpha_l^B)}{2B|cos(\beta_l^B)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.29}$$

Table 7.3: Revised Safety invariant formulas for 1D space model

clearly appears that $f^p_{crash}(l,f) = 1$ for the special configuration identified above as the intersection of both spatial entities is not null. In fact, we have $S(l) \cap S(f) = \left\{F_f^E\right\}$

3/2D and 3D space models. Objects in this approach are represented as respectively polygons and polyhedra of various shapes. Regular, well-defined shapes such as the rectangles in the right hand side of Figure 7.4 are in line with the 2D spatial representation models in Figure 4.3 while fitting the RCC-8 restrictions introduced in Section 4.4.2. Therefore, they are suitable but they are not always appropriate for the application at

hand. Accounting for all possible shapes of objects is impossible without some approximation. Therefore, we define and use the smallest regular bound (rectangle and polyhedra shape respectively) for the object. This allows the model to be conservative and precise enough without sacrificing the effectiveness of the prediction in the reasoning process (see *Remark 7.2*). On the other hand, in situations where the predicted "logical" collision does not happen, adversary conditions such the ones described in Section 6.3.1 may leave no way out for the objects to spatially separate as they will be already too close.

As for the previous case, all (restricted) RCC-compliant representations will be appropriate. The main constraint in the invariant is the absence of overlaps between the two shapes all the time. The application of the dynamic (stopping distance) constraint on the front and back of the objects as described above for 1D space model adds further restrictions to the model. Thus, we write the invariant formulas as follows (see Table 7.4). We obtain

$$(f^{p}_{crash}(l,f)=0)\,|_{2,3D} \wedge |X^F_f| + \frac{V_f^2 sin^2(\alpha^F_f) cos^2(\theta^F_f)}{2b|sin(\beta^F_f) cos(\lambda^F_f)|} < |X^B_l| + \frac{V_l^2 sin^2(\alpha^B_l) cos^2(\theta^B_l)}{2B|sin(\beta^B_l) cos(\lambda^B_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.30}$$

$$(f^{p}_{crash}(l,f)=0)\,|_{2,3D} \wedge |Y^F_f| + \frac{V_f^2 sin^2(\alpha^F_f) sin^2(\theta^F_f)}{2b|sin(\beta^F_f) sin(\lambda^F_f)|} < |Y^B_l| + \frac{V_l^2 sin^2(\alpha^B_l) sin^2(\theta^B_l)}{2B|sin(\beta^B_l) sin(\lambda^B_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.31}$$

$$(f^{p}_{crash}(l,f)=0)\,|_{2,3D} \wedge |Z^F_f| + \frac{V_f^2 cos^2(\alpha^F_f)}{2b|cos(\beta^F_f)|} < |Z^B_l| + \frac{V_l^2 cos^2(\alpha^B_l)}{2B|cos(\beta^B_l)|} \wedge V_f \geq 0 \wedge V_l \geq 0 \tag{7.32}$$

Table 7.4: Safety invariant formulas for 2D and 3D space models

the same formula as for 1D space, with the difference that objects here are either 2D or 3D spatial entities as indicated in the first constraint in the formulas. As stated before, only one of these 3 formulas needs to be satisfied to guarantee the separation, given that the right granularity of space has been used by the modeler. In other term, for a point representation in 3D of the object or any of its point, a unit displacement $\Delta d = 1$ on any of the 3 axes in any referential system and any direction is significant enough to ensure

spatial separation.

Remark 7.2 *(Tolerance for 2D and 3D representations).* Approximating the actual complex shape of a spatial entity in the world by a regular polygon or polyhedra bound without sacrificing the accuracy and precision, thus effectiveness of the reasoning process, requires not only to evaluate and track the tolerance of the spatial representation of objects but also the selection of the one that offers the better precision at the lowest computation cost. To that aim, we define the tolerance t_{dim} of the representation of a real world object S in dimension dim as the percentage of space of the object ADDED to its spatial model M. It's computed as follows.

In 2D,

$$t_{2D} = \frac{Area_M - Area_S}{Area_M} * 100 \tag{7.33}$$

In 3D,

$$t_{3D} = \frac{Volume_M - Volume_S}{Volume_M} * 100 \tag{7.34}$$

The smaller the value of t_{dim} the better the representation.

7.3.4 Local Lanes Safety Metrics for Away Collision

In this section, we build from lessons learned in Section 6.3.2, as well as results obtained in Section 7.3.3 and previous research to define and represent safety metrics for the away collision for the local lane control problem. Looking at the behavior of both the leader and the follower in Figure 7.3, it appears that, for the system to be always safe, constraint $Safe_\varepsilon$ has to be satisfied all the time. The satisfaction of this (set of) condition(s) - especially in case the follower is too close - could result in accident even if

the follower applies the maximum braking force. Therefore, we would like condition (7.1) to be satisfied all the time. We revisit the invariant relations uncovered in Section 7.3.3 to define the appropriate metrics for each situation. Given that the safety formula has to be true on only one of the axis we'll use the expression on the `x-axis` in the graphics for the sake of simplification of representations. However, all metrics for all axes must be computed and the conditions checked for all axes before drawing any conclusion on the safety of the system. Also, to keep the definition of metrics simple, we'll consider the conditions on the speed i.e. $V_f \geq 0 \wedge V_l \geq 0$ always verified.

In order for the system to be safe, the following three conditions have to be satisfied: (C1) relation (7.1) has to be verified in the (R) coordinate system then, (C2) the positions of the objects have to be separated on at least 1 of the coordinates and, (C3) their expected positions, should they brake in the worse case configuration, are also separated on at least one of the axes.

1/0D-based safety metrics. Using equation (7.1) and the transformation (T) along with the subsequent notations and calculations, we define the *Safe following metrics* Δ_{sf_ε} for the three coordinates as follows (see Table 7.5).

In order to satisfy (C1), either one of the following conditions should be met.

$$\Delta_{sf_\varepsilon}(l,f)|_{0D} \begin{cases} \Delta^X_{sf_\varepsilon} < 1 \\ \Delta^Y_{sf_\varepsilon} < 1 \\ \Delta^Z_{sf_\varepsilon} < 1 \end{cases} \tag{7.38}$$

Also, we define the *Longitudinal separation metrics* $\Delta_{ls}(l,f)$ as follows.

$$\Delta^X_{ls}(l,f)|_{0D} = \frac{|X_f|}{|X_l|} \tag{7.39}$$

$$\Delta^X_{sf_\varepsilon}(l,f)|_{0D} = \frac{|X_f| + \frac{V_f^2 sin^2(\alpha_f) cos^2(\theta_f)}{2b|sin(\beta_f)cos(\lambda_f)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f |sin(\alpha_f)cos(\theta_f)|\right)}{|X_l| + \frac{V_l^2 sin^2(\alpha_l) cos^2(\theta_l)}{2B|sin(\beta_l)cos(\lambda_l)|}} \tag{7.35}$$

$$\Delta^Y_{sf_\varepsilon}(l,f)|_{0D} = \frac{|Y_f| + \frac{V_f^2 sin^2(\alpha_f) sin^2(\theta_f)}{2b|sin(\beta_f)sin(\lambda_f)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f |sin(\alpha_f)sin(\theta_f)|\right)}{|Y_l| + \frac{V_l^2 sin^2(\alpha_l) sin^2(\theta_l)}{2B|sin(\beta_l)sin(\lambda_l)|}} \tag{7.36}$$

$$\Delta^Z_{sf_\varepsilon}(l,f)|_{0D} = \frac{|Z_f| + \frac{V_f^2 cos^2(\alpha_f)}{2b|cos(\beta_f)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f |cos(\theta_f)|\right)}{|Z_l| + \frac{V_l^2 cos^2(\alpha_l)}{2B|cos(\beta_l)|}} \tag{7.37}$$

Table 7.5: 0D-based safe following metrics for away collision with $\lambda_l, \beta_l \notin \{k\pi, (2k+1)\pi/2\} k \in \mathbb{Z}$

$$\Delta^Y_{ls}(l,f)|_{0D} = \frac{|Y_f|}{|Y_l|} \tag{7.40}$$

$$\Delta^Z_{ls}(l,f)|_{0D} = \frac{|Z_f|}{|Z_l|} \tag{7.41}$$

In order to satisfy (C2), one of the following relations should be satisfied by the metrics.

$$\Delta_{ls}(l,f)|_{0D} \begin{cases} \Delta^X_{ls} < 1 \\ \Delta^Y_{ls} < 1 \\ \Delta^Z_{ls} < 1 \end{cases} \tag{7.42}$$

Finally, the *Braking separation metrics* $\Delta_{bs}(l,f)$ are defined as follows.

$$\Delta^X_{bs}(l,f)|_{0D} = \frac{|X_f| + \frac{V_f^2 sin^2(\alpha_f) cos^2(\theta_f)}{2b|sin(\beta_f)cos(\lambda_f)|}}{|X_l| + \frac{V_l^2 sin^2(\alpha_l) cos^2(\theta_l)}{2B|sin(\beta_l)cos(\lambda_l)|}} \tag{7.43}$$

$$\Delta^Y_{bs}(l,f)|_{0D} = \frac{|Y_f| + \frac{V_f^2 sin^2(\alpha_f) sin^2(\theta_f)}{2b|sin(\beta_f)sin(\lambda_f)|}}{|Y_l| + \frac{V_l^2 sin^2(\alpha_l) sin^2(\theta_l)}{2B|sin(\beta_l)sin(\lambda_l)|}} \tag{7.44}$$

$$\Delta^Z_{bs}(l,f)|_{0D} = \frac{|Z_f| + \frac{V_f^2 cos^2(\alpha_f)}{2b|cos(\beta_f)|}}{|Z_l| + \frac{V_l^2 cos^2(\alpha_l)}{2B|cos(\beta_l)|}} \tag{7.45}$$

Satisfying (C3) requires that one of the following relations be satisfied by the metrics.

$$\Delta_{bs}(l,f)|_{0D} \begin{cases} \Delta_{bs}^{X} < 1 \\ \Delta_{bs}^{Y} < 1 \\ \Delta_{bs}^{Z} < 1 \end{cases} \tag{7.46}$$

Together, these 3 groups of safety metrics define the conditions under which the system will be safe using 0D space representation in a 3D coordinates system. Figure 7.5 (a) shows a synthesized view of how they interact to created a *safety tube* within which the system is safe. Outside of that tube, the system is unsafe.

Building the safety tube. Let's consider the variable Δ_m^{dim} which denotes the selected value calculated for a metric $m \in \{bs, ls, sf_\varepsilon\}$ in space of dimension dim (=0 in the figure). The procedure to create the tube (actually a cube) proceeds as follows.

- (S1) Pick any metric m then, compute all values of the metric Δ_m that need to be computed (i.e. sub metrics Δ_m^X, Δ_m^Y and/or Δ_m^Z) and, if none of the value is strictly smaller than 1 as stated in either equation (7.38), (7.42) or (7.46), then the system is unsafe. Otherwise, any of the value that satisfies the condition stated is assigned to Δ_m^{dim}.

- (S2) Repeat step (S1) for any of the next metrics $m^{'}$. If all sub metrics have values within the required range then, assign to $\Delta_{m^{'}}^{dim}$ variable and goto (S3), otherwise, the system is unsafe.

- (S3) Repeat step (S1) for the last metric $m^{"}$. If all sub metrics have values within the required range then, assign to the corresponding $\Delta_{m^{"}}^{dim}$. The corresponding point is represented within the tube meaning the system is safe otherwise, the system is unsafe.

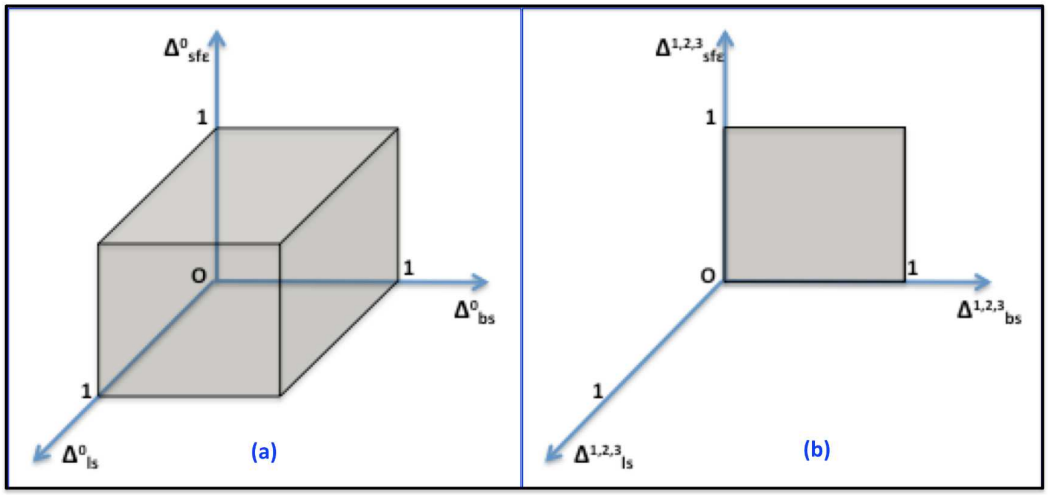


Figure 7.5: Safety tubes for local away collision control under : (a) 0D space models, (b) 1,2,3D space models .

2/1,2,3D-based safety metrics. For dimensions higher than 0, we need to distinguish the location of the front and back of respectively the follower and the leader in the expression of the *safe following metric* Δ_{sf_ε} in Table 7.5. It results the expressions in Table 7.6.

$$\Delta^X_{sf_\varepsilon}(l,f)|_{1,2,3D} = \frac{|X_f^F| + \frac{V_f^2 sin^2(\alpha_f^F)cos^2(\theta_f^F)}{2b|sin(\beta_f^F)cos(\lambda_f^F)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f|sin(\alpha_f^F)cos(\theta_f^F)|\right)}{|X_l^B| + \frac{V_l^2 sin^2(\alpha_l^B)cos^2(\theta_l^B)}{2B|sin(\beta_l^B)cos(\lambda_l^B)|}} \tag{7.47}$$

$$\Delta^Y_{sf_\varepsilon}(l,f)|_{1,2,3D} = \frac{|Y_f^F| + \frac{V_f^2 sin^2(\alpha_f^F)sin^2(\theta_f^F)}{2b|sin(\beta_f^F)sin(\lambda_f^F)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f|sin(\alpha_f^F)sin(\theta_f^F)|\right)}{|Y_l^B| + \frac{V_l^2 sin^2(\alpha_l^B)sin^2(\theta_l^B)}{2B|sin(\beta_l^B)sin(\lambda_l^B)|}} \tag{7.48}$$

$$\Delta^Z_{sf_\varepsilon}(l,f)|_{1,2,3D} = \frac{|Z_f^F| + \frac{V_f^2 cos^2(\alpha_f^F)}{2b|cos(\beta_f^F)|} + \left(\frac{A}{b}+1\right)\left(\frac{A}{2}\varepsilon^2 + \varepsilon V_f|cos(\theta_f^F)|\right)}{|Z_l^B| + \frac{V_l^2 cos^2(\alpha_l^B)}{2B|cos(\beta_l^B)|}} \tag{7.49}$$

Table 7.6: 1,2,3D-based safe following metrics for away collision, with $\lambda_l, \beta_l \notin \{k\pi, (2k+1)\pi/2\}$, $k \in \mathbb{Z}$

As for the previous case, in order for condition (C1) to be satisfied, constraints (7.38) need

to be applied to equations in Table 7.6 as follows.

$$\Delta_{sf_\varepsilon}(l,f)|_{1,2,3D} \begin{cases} \Delta^X_{sf_\varepsilon} < 1 \\ \Delta^Y_{sf_\varepsilon} < 1 \\ \Delta^Z_{sf_\varepsilon} < 1 \end{cases} \tag{7.50}$$

In the absence of a projection on individual axes, the *longitudinal separation* metric $\Delta_{ls}(l,f)$ is defined based on the crash function introduced in section D.2 and equations in Tables 7.3 and 7.4 as follows.

$$\Delta_{ls}(l,f)|_{1,2,3D} = f^p_{crash}(l,f) \tag{7.51}$$

For condition (C2) to be satisfied in this case, the following relation should be true all the time.

$$\Delta_{ls}(l,f)|_{1,2,3D} = 0 \tag{7.52}$$

We also need to distinguish the front and back locations of the leader and follower in defining the *braking separation* metric $\Delta_{bs}(l,f)$ as follows.

$$\Delta^X_{bs}(l,f)|_{1,2,3D} = \frac{|X^F_f| + \frac{V_f^2 sin^2(\alpha_f^F) cos^2(\theta_f^F)}{2b|sin(\beta_f^F) cos(\lambda_f^F)|}}{|X^B_l| + \frac{V_l^2 sin^2(\alpha_l^B) cos^2(\theta_l^B)}{2B|sin(\beta_l^B) cos(\lambda_l^B)|}} \tag{7.53}$$

$$\Delta^Y_{bs}(l,f)|_{1,2,3D} = \frac{|Y^F_f| + \frac{V_f^2 sin^2(\alpha_f^F) sin^2(\theta_f^F)}{2b|sin(\beta_f^F) sin(\lambda_f^F)|}}{|Y^B_l| + \frac{V_l^2 sin^2(\alpha_l^B) sin^2(\theta_l^B)}{2B|sin(\beta_l^B) sin(\lambda_l^B)|}} \tag{7.54}$$

$$\Delta^Z_{bs}(l,f)|_{1,2,3D} = \frac{|Z^F_f| + \frac{V_f^2 cos^2(\alpha_f^F)}{2b|cos(\beta_f^F)|}}{|Z^B_l| + \frac{V_l^2 cos^2(\alpha_l^B)}{2B|cos(\beta_l^B)|}} \tag{7.55}$$

As previously stated, those metrics are fully defined for $\lambda_l, \beta_l \notin \{k\pi, (2k+1)\pi/2\}$, $k \in \mathbb{Z}$.

Satisfying (C3) requires that one of the following relations be satisfied by the safe braking metrics.

$$\Delta_{bs}(l,f)|_{1,2,3D} \begin{cases} \Delta_{bs}^{X} < 1 \\ \Delta_{bs}^{Y} < 1 \\ \Delta_{bs}^{Z} < 1 \end{cases} \tag{7.56}$$

Together, constraints (7.50), (7.52) and (7.56) define the "volume" within which the system is safe. Using the variable Δ_{m}^{dim} introduced above, we represent on Figure 7.5 (b) this safety tube for the system when $dim \in \{1,2,3\}$. We note here that the null (0) value required for the metric $\Delta_{ls}(l,f)$ flattens the tube into a *safety square*, i.e. a 2D-shape.

Remark 7.3 In the case of the 1D space, the definition of the *safe following metric* assumes the use of the updated version of the safety invariant formulas in Table 7.3. If one uses the original version i.e. Table 7.2, a tube similar to the one in Figure 7.5 a) will be obtained instead.

7.4 Collision Avoidance Strategies and Algorithms

7.4.1 Generic Collision Avoidance Process

Traditional collision avoidance strategies decompose the process into three generic steps as follows [147].

1. **Sense:** Sensors such as radars or transponders are used to perform an accurate surveillance of potential targets/obstacles. The sensors can be standalone or organized into a coordinated, synchronized network to maximize efficiency.

2. **Detect:** Data collected from sensors are processed for the purpose of the determination as to whether a risk of collision or conflict exists and a characterization of the

risk (prediction) is established.

3. **Avoid:** The determination, characterization and execution of the appropriate avoidance maneuver for the risk (type of collision) identified are performed at this step.

Ensuring the safety of real-time systems such as the ones in the class of CPS we've been studying in this research requires this process be ran continuously as long as the system is operational. Also, it must be executed in little to no latency. This is not always guarantee as the last step could require significant actuation time and the first two might require extensive computation (algorithm execution), which consumes time too [32]. Moreover, the performance of the implementation (physical) platform can greatly affect the outcome of the process. In the context of this work, we introduce the generic Algorithm 1 as a foundation for further collision type-driven avoidance strategies.

Algorithm 1 Generic Safety Procedure for Safety-critical CPS

Input: Set of system parameters P; Set of acceptable safety range values $R_C = [r_j^C]$, $j \in \{1, \ldots, n_R\}$ for the problem's n_R applicable dimensionless metrics under a given set of hard constraints C

Output: Realization of the safety state $s \in \{safe, unsafe\}$ of the system S under constraints C

1: Take and save the (sub)set of sensor measurements X in physical space $\mathcal{X}$, ▷ Only relevant sensor readings are needed
2: Compute and save set of dimensionless safety metrics: $Pi_C = [\pi_j^C] = f(\text{X, P})$, $j \in \{1, \ldots, n_R\}$ ▷ Function f will encompass the transformations required by DA (if needed)
3: $idx \leftarrow n_R$ ▷ Variable storing the last "safe" index
4: $j \leftarrow 1$
5: **while** ($\pi_j^C \in r_j^C$ and $j \leq n_R$) **do** ▷ Search index for which any constraint $c \in C$ is violated
6: $\quad idx \leftarrow j$
7: $\quad j \leftarrow j + 1$
8: **end while**
9: **if** $idx < n_R$ **then** ▷ Establishing the safety state of the system
10: $\quad s \leftarrow unsafe$
11: **else**
12: $\quad s \leftarrow safe$
13: **end if**
14: **return** s ▷ The system safety state is s

In Algorithm 1, the focus has been on the first two steps of the above-described generic process. Sensing is covered in `line #1` while lines `#2` to `#15` determine whether the system is safe or not. We also note that the set of acceptable safety range values R_C is determined by analyses such as the ones performed in Sections 6.3 and 7.3. The last step on corrective action is left out of the algorithm because it's highly dependent on the configuration and capabilities of the system at the time the unsafe state is established. Also, we recognize the assumption the algorithm makes on the existence of formulae to compute the various safety metrics which is not always obvious to uncover as we've seen throughout this work.

7.4.2 Local Away, Glancing and Clipping Collision Avoidance Algorithms

In the context of collision avoidance, we side with the authors in [80] who rightfully point out that there is no "one-size-fits-all" collision avoidance strategy. Differences in types of collision lead to difference in avoidance strategies. Spatial constraints on entity trajectories as well as their dynamics further reduce avoidance options at hand. The higher the degree of freedom of the entities involved, better and more are the avoidance options. Thus, in this section, we build from foundational work in Sections 7.4.1 and 7.3 as well as chapters 4 and 5 to develop collision avoidance algorithms for away, glancing and clipping collisions. In order to keep the complexity of the algorithms in check, we'll treat away collision independently from the others.

Away Collision Avoidance Algorithm. In order to develop this algorithm, we need to modify and customize Algorithm 1 using the set of safety metrics uncovered in Section 7.3.4. We also need to:

(a) Define the appropriate set of system parameters based on the assumptions in Section 7.3.1,

(b) Select the appropriate ontological commitment for spatial representation of dynamic objects and,

(c) Make a clear distinction between the leader (l) and the follower (f).

The main constraint is

$$c_{fl} = (f \ll l) \in C \tag{7.57}$$

i.e. the follower must always be behind the leader.

Algorithm 2 modifies and adapts Algorithm 1 to the local away collision avoidance problem. It relies on system sensor measurements and metrics developed in section 7.3.4 to determine the safety state of the system. The algorithm is decomposed in three phases, each addressing different stages of the solution.

In `Phase 1`, relevant system sensor measurements – mainly from individual dynamic entity's position, velocity and orientation – are read and stored. We note here that the location and identity of the sensors to be read depends on the ontological commitment for the spatial representation of these objects as defined by the input $dim \in \{0, 1, 2, 3\}$.

In `Phase 2`, the system set of safety metrics is computed using the appropriate dimension-based formulas as determined in Section 7.3.4.

`Phase 3` uses the results of the metric computation and the set of acceptable safety range values to characterize the safety of the system.

Based on the returned value of s, corrective actions can be taken if needed. In the case $s = unsafe$, the range of possible corrective actions includes, depending on the entities allowed degree of freedom : (a) slow down the follower f to the speed of the leader l, (b)

Algorithm 2 Local away collision avoidance algorithm

Input: Set of system parameters $P = \{A, b, \varepsilon\}$; Ontological commitment for spatial representation of dynamic objects $dim \in \{0, 1, 2, 3\}$; Set of acceptable safety range values $R_C = \bigcup_{j=0}^{n_R-1}\{[0, 1[\}$ for the problem's $n_R = 9$ applicable dimensionless submetrics $\Delta_m^{coord}(l, f)|_{dim}$ (once dim is fixed), under the set of hard constraints $C = \{c_{fl}\}$; with $m \in \{bs, ls, sf_\varepsilon\}$ and $coord \in \{X, Y, Z\}$

Output: Realization of the safety state $s \in \{safe, unsafe\}$ of the system S under C

Phase 1 - (Sub)set system dynamics sensor measurements $\hat{\mathbf{X}}$

if $dim = 0$ **then**
  $\hat{\mathrm{X}}^f \leftarrow \{X_f, Y_f, Z_f, V_f, \alpha_f, \theta_f, \beta_f, \lambda_f\}$ ▷ Position, velocity and orientation follower
  $\hat{\mathrm{X}}^l \leftarrow \{X_l, Y_l, Z_l, V_l, \alpha_l, \theta_l, \beta_l, \lambda_l\}$, ▷ Position, velocity and orientation leader
else
  if $dim \geq 1$ **then**
    $\hat{\mathrm{X}}^f \leftarrow \{X_f^F, Y_f^F, Z_f^F, V_f, \alpha_f^F, \theta_f^F, \beta_f^F, \lambda_f^F\}$ ▷ Front (F) of follower (f)
    $\hat{\mathrm{X}}^l \leftarrow \{X_l^B, Y_l^B, Z_l^B, V_l, \alpha_l^B, \theta_l^B, \beta_l^B, \lambda_l^B\}$ ▷ Back (B) of leader (l)
  end if
end if
$\hat{\mathrm{X}} \leftarrow \hat{\mathrm{X}}^f \cup \hat{\mathrm{X}}^l$ ▷ System measurements

Phase 2 - Set of safety metrics: $[\Delta_m^{coord}(l, f)|_{dim}] = [\pi_j^C] = f(\hat{\mathbf{X}}, \mathbf{P})$, $j \in \{0, \ldots, 8\}$

Create empty list of π_j^C: Pi_C
for coord $\in \{X, Y, Z\}$ **do**
  for m $\in \{bs, ls, sf_\varepsilon\}$ **do**
    if $dim = 0$ **then**
      Compute $\Delta_m^{coord}(l, f)|_{0D}$ ▷ Use eqs. 7.35 – 7.37, 7.39 – 7.41, and 7.43 – 7.45
    else
      Compute $\Delta_m^{coord}(l, f)|_{1,2,3D}$ ▷ Use eqs. 7.47 – 7.49, 7.51, and 7.53 – 7.55
    end if
    Add result to list: $Pi_C \leftarrow \Delta_m^{coord}(l, f)|_{dim}$
  end for
end for

Phase 3 - Characterization of system safety

$idx \leftarrow |Pi_C| - 1$ ▷ Variable storing the last "safe" index; $n_R = |Pi_C|$
$j \leftarrow 0$
while ($\pi_j^C \in [0, 1[$ and $j < |Pi_C|$) **do** ▷ $\pi_j^C \in Pi_C$ and $r_j^C = [0, 1[, \forall j \in \{0, \ldots, 8\}$
  $idx \leftarrow j$
  $j \leftarrow j + 1$
end while
if $idx < |Pi_C| - 1$ **then** ▷ Establishing the safety state of the system
  $s \leftarrow unsafe$
else
  $s \leftarrow safe$
end if
return s ▷ The system safety state is s

move l forward faster or, (c) move l away (left, right, up, down) from the trajectory of f [80].

Glancing and Clipping Collision Avoidance Algorithm. One key advantage of safety metrics is their ability to encapsulate spatial and temporal properties of the system into important, powerful system-level dimensionless parameters for safety modeling and analysis of safety-critical systems such as CPS. This has been proven very effective in the case of the away collision algorithm. However, as of now, we don't have predefined metrics to assist in detecting and characterizing glancing or clipping collisions. Therefore, developing avoidance algorithms is an effort that requires detail knowledge of spatial and temporal representations and constraints applicable to the system. Thus, we'll build on semantically supported metadomain representations developed throughout this work as part of the CPS-KMoDS. This includes the temporal and spatial modeling system introduced respectively in Chapters 3 and 4.

Thus, we need to define and assign :

(a) (Intersecting) travel trajectories to individual moving entities and,

(b) Make a clear distinction between the entities $id \in 1, 2$ and their priority in term of right of way (ROW), predefined traffic rules.

In light with the multi-level spatial representation scheme introduced in Section 4.4.3, we introduce the following definition.

Definition 7.4.1. *(Trajectory)* A trajectory is a sequence of adjacent spatial entities of identical dimensions. It can be closed or open but also oriented. In order words, $Tr(i)|_{dim} = \bigcup_{t=0}^{n-1}\{s_t^i\}|_{dim}$, with s_t^i spatial entities, $dim \in \{0, 1, 2, 3\}$, and, $n \geq 1$ is a trajectory if all of the following conditions are satisfied: (i) Each s_t^i is either a Primitive, Extended or Composite entity (as defined in Section 4.4.3), (ii) All entities' dimension is

the same as the one of the trajectory $Tr(i)$ i.e. $\forall s_t^i$, $dim(s_t^i) = dim_{Tr(i)}$ and, (iii) Two consecutive entities in $Tr(i)$ are adjacent i.e. $\forall t \in \{0, \ldots, n-1\}$, $s_{t+1}^i \in Adj(s_t^i)$ (when $n \geq 2$).

When the trajectory is positively oriented, it's noted $Tr^+(i)$ and its spatial entities are traversed in the ascendant order of occurrence in the list. This order is reverse when it's negatively oriented and noted $Tr^-(i)$. For the remaining part of this document, we'll assume trajectories are positively oriented unless explicitly indicated. One important consequence of this definition is that a trajectory can be represented at various levels of abstractions, using the appropriate spatial ontological commitment defined by the parameter dim. Also, the intersection of two trajectories of dimensions k and k' can be either null, or an (or set of) entity of dimension $k'' \leq min(k, k')$. Similarly, the spatial representation of (a dynamic) entity e has to be made complete by specifying its highest spatial dimension as $e|_{dim}$. For instance, when $dim = 2$, the geometry of e is a Polygon as illustrated on the right side of Figure 4.3. This also means that e can be represented at lower dimensions i.e. as a Line ($dim := 1$) or Point ($dim := 0$).

Remark 7.4 When the maximum dimension is not represented, the default dimension i.e. $dim = 2$ is assumed. We can picture the trajectory of $e|_{dimE}$ in dimension $dimS$ as its (expected) trace when it moves in a specific direction, when represented in that spatial dimension. However, the closest physical entity representation of the real world entity is better done with $dimE \in \{2, 3\}$.

Equipped with this definition, we make the following additional assumptions.

(B1) The lower the Id of the entity, the higher is its ROW.

(B2) Entities i and j travel at velocities within the allowed speed limits in each of the segments of trajectory $Tr^+(i)$ and $Tr^+(j)$ respectively.

(B3) Similarly to assumption (A3), the computation time is assumed to be negligible thus, will not affect safety procedures.

In order to support the development of the glancing and clipping collision avoidance algorithms, we define and introduce spatial and temporal intersection functions. Algorithm 3 describes the procedure through which the intersection of two trajectories expressed in various dimensions is computed. The algorithm returns either an empty set or a single spatial entity in a dimension $k'' \leq min(k, k')$. When the geometric intersection of component entities in the trajectories leads to multiple entities (`line #13`), the algorithm extracts the first subset of adjacent entities (`line #14`) – for instance if the trajectories cross each other several times – then, composes individual entities into a larger one of the same dimension (`line #17`). The composition ($MergeWith$) operation preserves the spatial decomposition procedure introduced in Section 4.4 thus, the consistency of the results of RCC-8 operations on the composed entities is guaranteed.

Dealing with time requires an extra effort. Thus, we introduce the function defined by Algorithm 4 that computes the intersection of two proper time intervals as per Allen's temporal interval calculus.

Given that (dynamic) entities are already annotated in definition 7.4.1 we can use the result to spatially annotate temporal entities as well, especially instants. Thus, $t_{s_t^i|_k}$ will denote the time instant associated to the event of entity i being at location or occupying spatial entity $s_t^i|_k$.

The last element we need is the set of safety constraints associated to the glancing and clipping collisions avoidance problem. Thus, we consider the spatial intersection $s^{ij}|_k$ of two trajectories of identical dimension k as computed by Algorithm 3 as follows : $s^{ij}|_k = Tr^+(i)|_k \cap Tr^+(j)|_k$; with $k \in \{0, .., 3\}$. For glancing collision, the main constraint

Algorithm 3 Spatial intersection of two trajectories dim of k & k' ($k, k^{'} \in \{0, .., 3\}$)

function SPATIALINTERSECTION($Tr^{+}(i)|_k = \bigcup_{t=0}^{n-1}\{s_t\}|_k$,$Tr^{+}(j)|_{k'} = \bigcup_{l=0}^{m-1}\{s_l\}|_{k'}$)
  $s^{ij}|_{k^{''}} \leftarrow \emptyset$ ▷ Result of the intersection $k^{''} \leq min(k, k^{'})$
  $TempList^{ij}|_{k^{''}} \leftarrow \emptyset$
  for $t \in \{0, .., n-1\}$ **do**
    for $l \in \{0, .., m-1\}$ **do**
      $Ovlp^{tl}|_{k^{''}} \leftarrow s_t|_k \cap s_l|_{k'}$ ▷ RCC-8 compliant geometric operation
      if $Ovlp^{tl}|_{k^{''}} \neq \emptyset$ **then**
        $TempList^{ij}|_{k^{''}} \leftarrow Ovlp^{tl}|_{k^{''}}$ ▷ Add to list
      end if
    end for
  end for
  $sizeList \leftarrow |TempList^{ij}|_{k^{''}}|$
  if $sizeList > 1$ **then**
    $adjSetList|_{k^{''}} \leftarrow Adj(TempList^{ij}|_{k^{''}})[0]$ ▷ Set of adjacent entities in the list
    $s^{M}|_{k^{''}} \leftarrow adjSetList|_{k^{''}}[0]$ ▷ Initialize merging entity
    for $p \in \{1, .., |adjSetList|_{k^{''}}| - 1\}$ **do**
      $s^{M}|_{k^{''}} \leftarrow MergeWith(adjSetList|_{k^{''}}[p])$ ▷ Geometric composition
    end for
    $s^{ij}|_{k^{''}} \leftarrow s^{M}|_{k^{''}}$
  else
    if $sizeList = 1$ **then**
      $s^{ij}|_{k^{''}} \leftarrow TempList^{ij}|_{k^{''}}[0]$ ▷ Save the only element in the list
    end if
  end if
  return $s^{ij}|_{k^{''}}$ ▷ The intersection can be an empty set $\emptyset$
end function

Algorithm 4 Intersection of two proper time intervals

function TEMPORALINTERSECTION($[t_1, t_2]$,$[t_3, t_4]$)
  $[t_b, t_e] \leftarrow \emptyset$ ▷ $t_b = beginInstant$ and $t_e = endInstant$ of resulting time interval
  if $t_3 \leq t_2$ **then**
    $t_b \leftarrow max(t_1, t_3)$
    $t_e \leftarrow min(t_2, t_4)$
  end if
  return $[t_b, t_e]$ ▷ The Intersection can be an empty set or an instant (if $t_b = t_e$)
end function

is

$$c_{ij}^{G} = (e_i \neq e_j, \quad \forall i,j/i \neq j \quad s^{ij}|_0 = \emptyset) \in C \tag{7.58}$$

i. e. ANY two (dynamic) entities picked among the family of n_E entities on glancing collision course must be separated spatially at all time in the lowest dimension ($s^{ij}|_0 = \emptyset$). Similarly, the main constraint for clipping collision can be written as follows.

$$c_{ij}^{C} = (e_i \neq e_j, \quad \forall i,j/i \neq j \quad s^{ij}|_2 = \emptyset) \in C \tag{7.59}$$

We note that, thanks to the possibility of multidimensional representations of entities and trajectories offered by our framework, we can clearly distinguish glancing and clipping collisions using the value of $s^{ij}|_k$. Therefore, we take advantage of this representation in Algorithm 5 to support the description of the procedure through which glancing and clipping collisions are predicted and resolved before they materialize.

Among the key inputs of the algorithm are the ontological commitment for spatial representation $dimE$ of a pair of dynamic objects $E = (e_i, e_j)|_{dimE}$ along with their trajectories $Tr(i)|_k$ and $Tr(j)|_k$ expressed in dimensions $k \in \{0, .., dimE\}$. Also, the set of system parameters P and hard constraints C are also needed. In `Phase 0`, algorithm variables are defined and initialized.

The goal of `Phase 1` is to research and predict, based on current configuration of the system and given inputs, any future spatial conflict and characterize the type of collision (if any). First, the intersection of spatial entities at all dimensions is computed and stored (`lines #11` to `#13`) using spatial function in Algorithm 3. Then, in `lines #15` to `#21`, the lowest dimension idx at which there is a spatial conflict is extracted (if any). As shown in Figure 7.6 the value of dimension idx is also used to differentiate possible glancing

Algorithm 5 Local glancing and clipping collision avoidance algorithm

Input: Pair of dynamic entities $E = (e_i, e_j)|_{dimE}$; Ontological commitment for spatial representation of dynamic objects $dimE \in \{2, 3\}$; Trajectory of entity i, $Tr(i)|_k = \bigcup_{t=0}^{n-1}\{s_t^i\}|_k$ and entity j, $Tr(j)|_k = \bigcup_{l=0}^{m-1}\{s_l^j\}|_k$, with $n, m \geq 2$, $i < j$ and $k \in \{0, .., dimE\}$; Set of system parameters $P = P^i \cup P^j \cup P^{space}$; Reaction velocity $v_\epsilon > 0$; Set of hard constraints $C = \{c_{ij}^G, c_{ij}^C\}$ applicable to the system.

Output: Realization of the safety state $s \in \{safe, unsafe\}$ of the system S under C; Type of collision (avoided) $colType \in \{none, glancing, clipping\}$

Phase 0 - Initialize variables

$sc \leftarrow false$ ▷ Initialize variable indicating whether there is a spatial conflict or not
$tc \leftarrow false$ ▷ — // — — // — — // — temporal — // —
$dimS \leftarrow dimE$
$idx \leftarrow dimS + 1$
$s^{ij}|_k \leftarrow \emptyset \ \forall k \in \{0, .., dimS\}$ ▷ First collision spatial entity for i and j in $dim = k$
$O^{ij}|_0 \leftarrow O|_0$ ▷ Closest collision POINT on $s^{ij}|_k$ for i and j
$A^i|_0, A^j|_0 \leftarrow O|_0$ ▷ Closest approach POINT on $Tr(i)$, $Tr(i)$ to $O^{ij}|_0$ for i,j
$colType \leftarrow none$
$t_b^{ij}|_0, t_e^{ij}|_0 \leftarrow 0$ ▷ Begin and end of intersection time interval; dimS =0
$s \leftarrow unsafe$ ▷ By default, the system is unsafe

Phase 1 - Predict spatial conflict and collision type based on trajectories

for $k \in \{0, .., dimS\}$ **do** ▷ Compute intersections for all possible dimensions
    $s^{ij}|_k \leftarrow SpatialIntersection(Tr(i)|_k, Tr(j)_k)$ ▷ Use Algorithm 3
end for
$k \leftarrow 0$
while ($k \leq dimS$ and $sc = false$) **do** ▷ Lowest dim. of non empty spatial intersect.
    if $s^{ij}|_k \neq \emptyset$ **then**
        $sc \leftarrow true$
        $idx \leftarrow k$ ▷ There is a spatial conflict for representations at $dim = idx$
    end if
    $k \leftarrow k + 1$
end while
if $sc = false$ **then**
    $s \leftarrow safe$ ▷ There is NO SPATIAL conflict, hence no risk of collision ahead
    return $s, colType$ ▷ The system is $s = safe$ and $colType = none \Rightarrow STOP$!
end if
$s^i|_2, s^j|_2 \leftarrow \emptyset$ ▷ Spatial variables
if $idx = 0$ or $idx = 1$ **then** ▷ GLANCING collision between trajectories
    $colType \leftarrow glancing$ ▷ See Figure 7.6 (a_0) to (c_0) and (a_1) to (b_1)
    $s^i|_2 \leftarrow s|_2 \in Tr(i)^+|_2 / SpatialIntersection(s|_2, s^{ij}|_2) = s^{ij}|_2$ ▷ Use Algorithm 3
    $s^j|_2 \leftarrow s|_2 \in Tr(j)^+|_2 / SpatialIntersection(s|_2, s^{ij}|_2) = s^{ij}|_2$
    $O^{ij}|_0 \leftarrow BL(s^{ij}|_2)|^{\theta ij}$ or $BR(s^{ij}|_2)|^{\theta ij}$ ▷ BL & BR as per Fig. 4.3; θij = orient. $s^{ij}|_2$
end if

33: **if** $idx = 2$ **then** ▷ There is a CLIPPING collision between the trajectories
34: $colType \leftarrow clipping$
35: **if** $dim(s^{ij}|_2) = 0$ **then** ▷ The trajectories meet at POINT $s^{ij}|_2$; See Figure 7.6 (c_2)
36: $s^i|_2 \leftarrow s|_2 \in Tr(i)^+|_2 / SpatialIntersection(s|_2, s^{ij}|_2) = s^{ij}|_2$ ▷ Use Algo. 3
37: $s^j|_2 \leftarrow s|_2 \in Tr(j)^+|_2 / SpatialIntersection(s|_2, s^{ij}|_2) = s^{ij}|_2$
38: $O^{ij}|_0 \leftarrow s^{ij}|_2$
39: **else**
40: **if** $dim(s^{ij}|_2) = 1$ **then** ▷ Trajects. meet at LINE $s^{ij}|_2$; See Fig. 7.6 (b_2)
41: $s^i|_2 \leftarrow s^i \in Tr^+(i)|_2 / SpatialIntersection(s^{ij}|_2, s^i) = Back(s^{ij}|_2)$
42: $s^j|_2 \leftarrow s^j \in Tr^+(j)|_2 / SpatialIntersection(s^{ij}|_2, s^j) = Back(s^{ij}|_2)$
43: $O^{ij}|_0 \leftarrow Back(s^{ij}|_2)$
44: **else**
45: **if** $dim(s^{ij}|_2) = 2$ **then** ▷ Traject. meet at POLYGON $s^{ij}|_2$; See Fig. 7.6 (a_2)
46: $s^i|_2 \leftarrow s^i \in Tr^+(i)|_2 / SpatialIntersection(s^{ij}|_2, s^i) = BL(s^i|_2)|^{\theta i} or BR(s^i|_2)|^{\theta i}$ ▷ BL or BR depends on θi
47: $s^j|_2 \leftarrow s^j \in Tr^+(j)|_2 / SpatialIntersection(s^{ij}|_2, s^j) = BL(s^j|_2)|^{\theta j} or BR(s^j|_2)|^{\theta j}$
48: $O^{ij}|_0 \leftarrow BL(s^{ij}|_2)|^{\theta ij} or BR(s^{ij}|_2)|^{\theta ij}$
49: **end if**
50: **end if**
51: **end if**
52: $A^i|_0 \leftarrow BL(s^i|_2)|^{\theta i} or BR(s^i|_2)|^{\theta i}$ ▷ BL & BR as per Fig. 4.3; θi= orientation i
53: $A^j|_0 \leftarrow BL(s^j|_2)|^{\theta j} or BR(s^j|_2)|^{\theta j}$
54: **end if**

Phase 2 - Predict Temporal conflict based on system dynamic

55: $\hat{\mathrm{X}}_{A^i|_0} \leftarrow \{A^i|_0, v_{A^i|_0}, t_{A^i|_0}\}$ ▷ Position, velocity and time stamp of i at location $A^i|_0$
56: $\hat{\mathrm{X}}_{A^j|_0} \leftarrow \{A^j|_0, v_{A^j|_0}, t_{A^j|_0}\}$ ▷ Position, velocity and time stamp of j at location $A^j|_0$
57: $t^i_{O^{ij}|_0} \leftarrow t_{A^i|_0} + dur(A^i|_0, O^{ij}|_0)$ ▷ Computed travel time $dur(A^i|_0, O^{ij}|_0) = \mathrm{f}(\hat{\mathrm{X}}_{A^i|_0}, P^i)$
58: $t^j_{O^{ij}|_0} \leftarrow t_{A^j|_0} + dur(A^j|_0, O^{ij}|_0)$ ▷ Expected arrival time at collision point
59: $[t^{ij}_b|_0, t^{ij}_e|_0] \leftarrow TemporalIntersection([t_{A^i|_0}, t^i_{O^{ij}|_0}], [t_{A^j|_0}, t^j_{O^{ij}|_0}])$ ▷ Use Algo.4
60: **if** $[t^{ij}_b|_0, t^{ij}_e|_0] = \emptyset$ or $t^{ij}_b|_0 \neq t^{ij}_e|_0$ **then**
61: $s \leftarrow safe$ ▷ There is spatial but NO TEMPORAL conflict
62: **return** $s, colType$ ▷ The system is $s = safe$ and $colType \in \{glancing, clipping\}$
63: **else**
64: $tc \leftarrow true$ ▷ There are spatial AND temporal conflict
65: **end if**

Phase 3 - Prevent collision by (time) separating back of i & front of j

66: $B^i|_0 \leftarrow BL(i)|_2|^{\theta i}$ or $BR(i)_2|^{\theta i}$ ▷ Back of entity i as per Fig. 4.3
67: $t_{BO^i|_0} \leftarrow t_{B^i|_0} + dur(B^i|_0, O^{ij}|_0)$ ▷ Expected arrival time of B^i at collision point
68: **while** $t_{BO^i|_0} > t^j_{O^{ij}|_0}$ **do**
69: $v_{A^j|_0} \leftarrow v_{A^j|_0} - v_\epsilon$ ▷ Slow down entity j (has lowest priority a per Assumption(B1))
70: $t^j_{O^{ij}|_0} \leftarrow t_{A^j|_0} + dur(A^j|_0, O^{ij}|_0)$ ▷ Update $t^j_{O^{ij}|_0}$; with $dur(A^j|_0, O^{ij}|_0) = \mathrm{f}(\hat{\mathrm{X}}_{A^i|_0}, P^i)$
71: **end while**

72: $tc \leftarrow false$ ▷ The temporal conflict has been resolved
73: $s \leftarrow safe$
74: **return** $s, colType$ ▷ The system is $s = safe$ and $colType \in \{glancing, clipping\}$

collision $idx = 0, 1$ (`line #28`) from clipping collision $idx = 2$ (`line #33`) in Algorithm 5. When $idx = 0$ the trajectories intersect at their lowest spatial dimension of representation ($dimS^{ij}|_0 = 0$) and we have the configuration (c_0). However, finding the system closest collision point $O^{ij}|_0$ and the two closest approach points $A^i|_0$ and $A^j|_0$, requires to trace the spatial representation of the trajectories all the way up to the one at dimension 2 as shown by the arrows between (c_0) and (a_0) via (b_0). Similarly, the traceability between representations (b_1) and (a_1) shows the mean through which closest collision and approach points are determined when $idx = 1$ and $dimS^{ij}|_1 = 0$. In case $dimS^{ij}|_1 = 1$, we have a configuration similar to (b_0). When $idx = 2$, the three ways the entities can clip are illustrated in configurations (a_2), (b_2) and (c_2). Finally, through the investigation of the dimension of the intersecting entity $s^{ij}|_{idx}$, we compute and store both the closest collision POINT on $s^{ij}|_k$ and approach POINT on $Tr(i)$, $Tr(i)$ to $O^{ij}|_0$ for i and j (`lines #27` to `#54`).

`Phase 2` makes use of Algorithm 4 and system dynamic to compute the intersection of temporal intervals made of the time instants at approaching points $t_{A^i|_0}$ and $t_{A^j|_0}$ and the expected time instants at collision point $t^i_{O^{ij}|_0}$ and $t^j_{O^{ij}|_0}$. The result is used to predict any temporal conflict between the dynamic entities.

The goal of `Phase 3` is to resolve the spatio-temporal conflict by separation of the colliding entities in the temporal domain. Thus, we make sure that the back of the entity with the highest traffic priority i.e. i will cross the collision point $O^{ij}|_0$ BEFORE the front of the entity of lowest priority j (`lines #68` to `#71`). This is done by slowing down entity j via

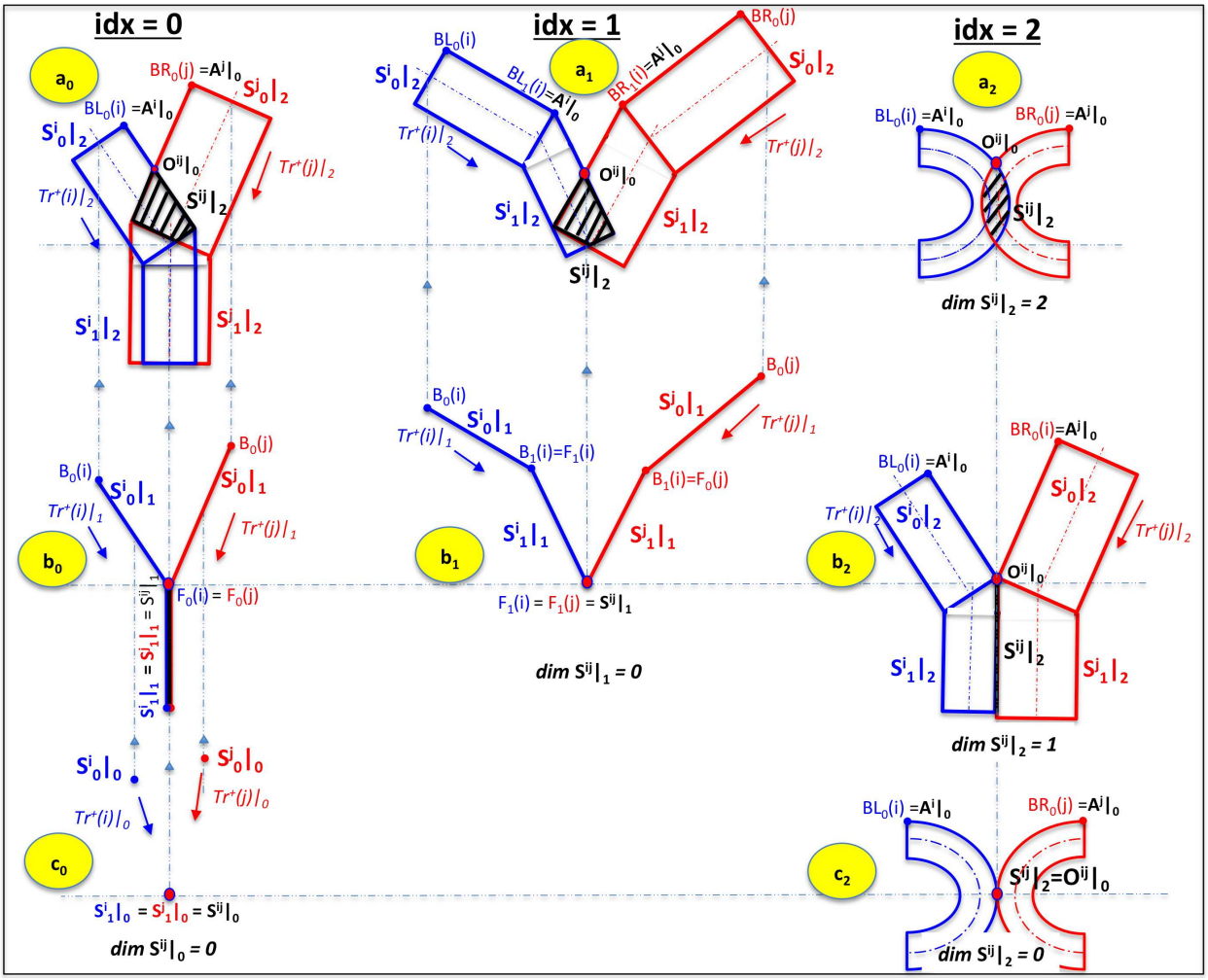


Figure 7.6: Representations of collision configurations for spatio-temporal algorithms

a speed reduction (braking) as shown in `line #69`.

Remark 7.5 Algorithms 2 and 5 address away, glancing and clipping collision avoidance strategies. Achieving the "zero accident" goal indicated in Chapter 4 requires the various actors to make the right decision at the right time but also the system as a whole to apply the right (combination of) collision avoidance strategy(ies) for the situation at hand. This adds another layer of complexity on the implementation of the algorithms.

7.5 Case Study: Glancing Collision at Non-signalized Intersection

7.5.1 Overview of the case study

We consider the problem of a glancing collision between two smart cars at a non-signalized traffic intersection. All the stoplights at the intersection are replaced by a single intersection manager (IM) that monitors traffics, establishes collision risks (if any) and intervenes to resolve spatio-temporal conflicts impeding safe crossings at the intersection. We seek to understand how the ontological commitment of the spatial model of the vehicles and their representation affects the outcome of the reasoning process for collision prediction. To that aim, we consider a particular but simple configuration of the modeling landscape where reasoning for collision avoidance is done with spatial representations at $dimE \in \{0, 1\}$ and $dimS = 1$. The problem is approached from a CPS perspective in the sense that vehicles are dynamic (physical) objects equipped with sensing, computation and communication (cyber) capabilities that communicate and interact with each other and the IM. For the purpose of this experiment, we assume that both vehicles travel at constant but different speeds and they are within sensing range of each other. We have developed a Java-based software platform consistent with the simulation architecture in Figure 6.4 and interprets the semantic network of 2D spaces adopted by the Open Street Map (OSM) community [197]. As such, the spatial models used in this experiment are at level L2 and lower on the hierarchy of spatial modeling (see Figure 4.3).

With the help of JavaFX, we created and visualized a race track (full details not shown) that preserves the semantic information of space. The lower right-hand of Figure 4.4 shows a zoom on the intersection of the track of interest for our application.

7.5.2 Spatio-temporal reasoning for glancing collision avoidance

We rely on the detailed architecture in Figure 4.4 to drive the implementation of the spatio-temporal reasoning scheme. We consider here a scenario involving a glancing collision (as pictured in Figure 7.1) between two driverless vehicles.

1/Component Models. In order to keep the experiment simple yet explicit enough to maintain the focus on the topic of this Chapter, we consider only 2 vehicles operating in the limited space representing the intersection. Thus, the vehicles are "dynamic" components and the intersection itself is considered a "static" component. Both component types have non-spatial features as illustrated in Section 4.4.3. However, each vehicle is assigned a predefined trajectory, both intersecting at the location `s13` inside the space occupied by the intersection. Control points are located on the track, at trajectories intersection or curvatures to keep track of the distance of the vehicle to eventual/candidate conflict areas.

2/Spatial Models. Each of the component type within the system has a spatial extension. As a case in point, the traffic intersection is modeled as an "IrregularSpaceBlock," which is an extended spatial entity in layer L2 in the hierarchical model on Figure 4.3. Figure 7.7 illustrates the XML representation of the `intersection1` as an irregular space block. The geometry is a JTS encoded polygon that defines the precise contour of the intersection as an ordered list of JTS points (in 0D). Also, points of interest (pois) as well as metrics (e.g., area) and features (e.g., name, identifier) can be captured by the model. The dynamic nature of vehicles along with the expected use of its spatial model for reasoning purpose pose a challenge on the choice of the appropriate level of spatial representation needed as explained in Section 4.4.3. This choice affects the effectiveness of the reasoning. For this

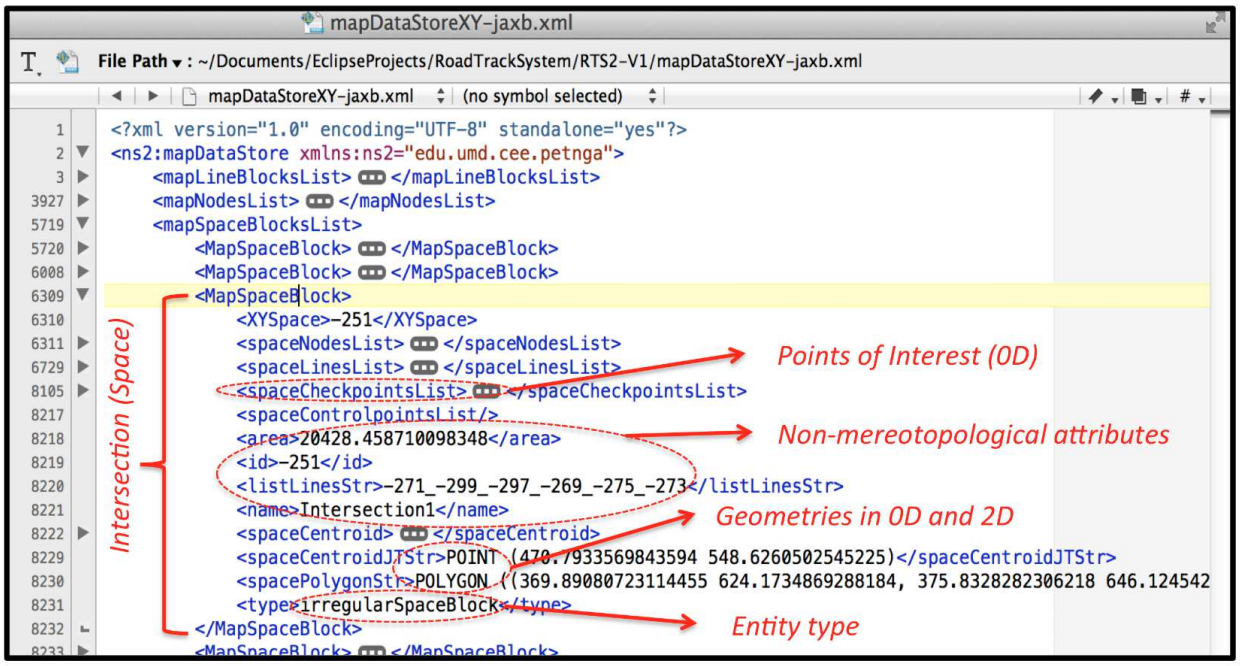


Figure 7.7: Data view of the model of an intersection as an irregular space block in XML.

experiment, a vehicle i will be viewed either as a:

(a) 2D Point which is the centroid $G(i)$ of its shape in 2D,

(b) Straight Line connecting its front $F(i)$ to its back $B(i)$ or,

(c) Polygon (rectangle) represented by its corner points as shown in Figure 4.3.

These geometries correspond respectively to Node (L0), LineBlock (L1) and RegularSpace-Block (L2) spatial models. We add position sensors (0D) at those points of interest on the vehicle boundary to track their position in real-time during simulation.

3/Spatio-temporal Reasoning. Effective collision avoidance requires the separation of entities in temporal and spatial domains. Therefore, the reasoner - embedded in the IM - should implement Algorithm 5 in order to achieve that goal. However, considering the goal of this analysis, we'll conduct a special implementation, with spatial representations at $dimE \in \{0, 1\}$ and $dimS = 1$.

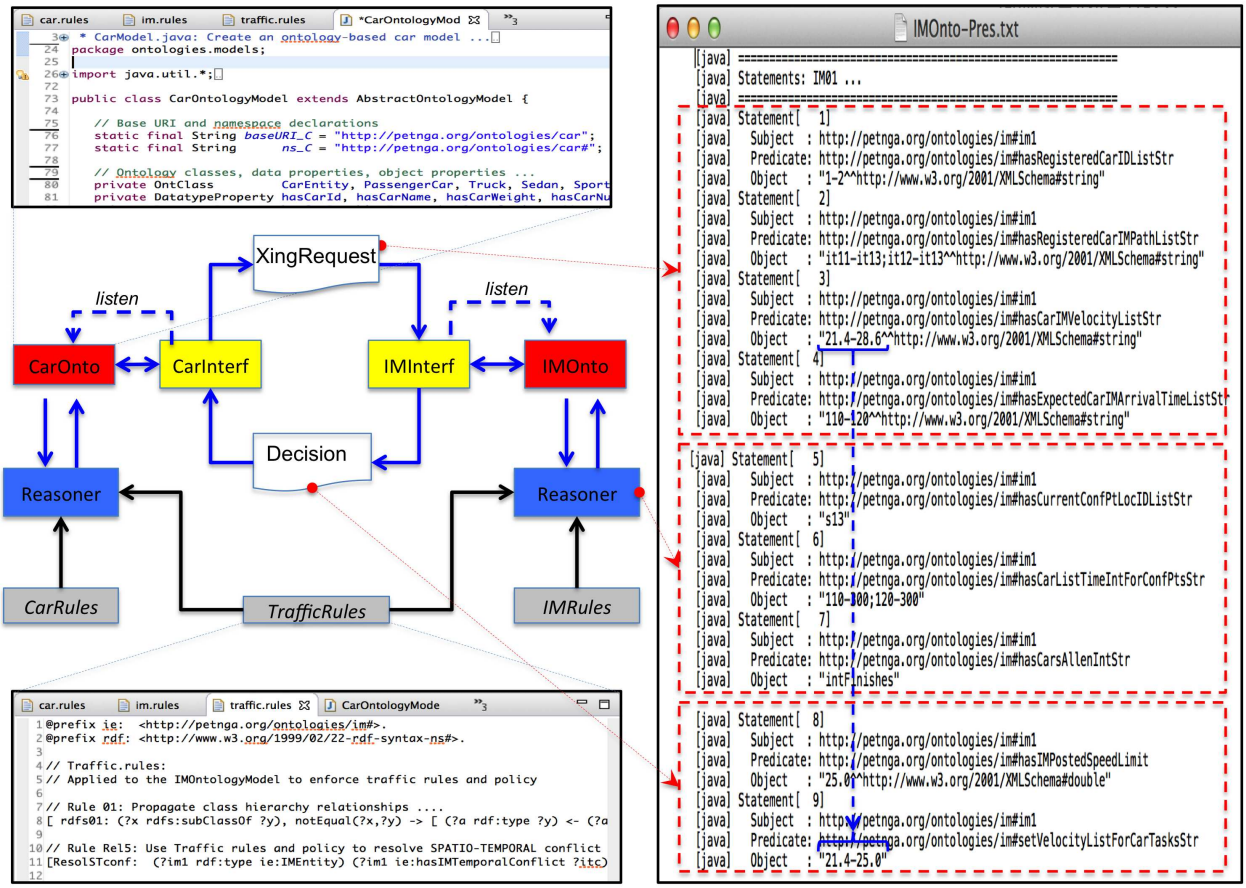


Figure 7.8: Illustration of ontology-based communication and control for spatio-temporal reasoning in a distributed traffic system architecture.

A collision is predicted to occur when the spatial representations for vehicles V(1) and V(2) – let's call them S1 and S2 – are predicted to occupy the same location at the some point in time. In other words, one of the RCC-8 spatial predicates $PO(S1, S2)$ or $EC(S1, S2)$ will evaluate to true. We assign a path to each vehicle on the racetrack and a constant but different traveling speed to each of them. With $dimE = 0$ and $dimS = 1$, the system is modeled as two 0D dots (vehicles) traveling along intersecting tracks (trajectories), which would correspond to a configuration similar to (b_1) in Figure 7.6. However, when $dimE = 1$ the system is configured as a 1D line (with a front and back as shwon in Figure 4.3) traveling along another line(s). After the set up, the simulation

is launched.

The left side of the Figure 7.8 illustrates the implemented ontology-based communication and control for spatio-temporal reasoning for the traffic system viewed as a distributed system. Each domain ontology is extended by its reasoner implementing domain rules. System level rules (traffic rules) are applied to all entities. Ontologies communicate via their interface which listen to (and communicate relevant) changes in the semantic graph across the domains. Vehicles send "crossing request" containing required information (`Statements 1 - 4`) to the IM and it returns its decision on conflict resolution (`Statements 8 - 9`). The right hand side of the Figure illustrates what is happening behind the scene as the collision is predicted and resolved before it materializes in the physical world.

In the first phase of the reasoning algorithm, the IM computes the spatial intersection of the two trajectories which yields `s13` (i.e. $O^{ij}|_0$ in Algorithm 5) as the intersection location. Given the dimension ($dim(s_{13}) = 1$) of this spatial entity and the classification of collisions in Figure 7.1, the IM predicts a glancing collision between the two vehicles at that specific location. The right hand side of the Figure shows that for the reasoning process to be effective and decidable, each vehicle needs to register to the IM (`Statement 1`), communicate its trajectory (`Statement 2`), current velocity (`Statement 3`) and expected arrival time at the closest approach point (i.e. $A^i|_0$ and $A^j|_0$ in Algorithm 5) to the control zone (`Statement 4`).

In the second phase, the reasoner computes the travel duration of each vehicle to the collision point (as stated in `lines #57` and `#58` of the algorithm) and infers the corresponding time interval of each vehicle while in the intersection control zone (`Statement 6`). Then, the computation of the temporal intersection of the two time intervals estab-

lishes the nature of the relationship between them as per Allen's temporal interval calculus (`Statement 7`). The "intFinishes" result clearly indicates a temporal conflict between the two intervals as shown by the corresponding configuration in Figure 3.3. This knowledge - combined with the one on the already established spatial conflict at `s13` - completes the prediction of the glancing collision at that specific location.

The collision (i.e. spatio-temporal conflict) is resolved (in the cyber world) and prevented (in the physical world) in phase 3 of the reasoning algorithm. In order to realize this goal, the IM identifies the vehicle with the lowest traffic priority (i.e. `#2`) and compute the velocity required to achieve the temporal separation at location `s13` in accordance with traffic rules such as the posted speed limit for that branch of the trajectory (`Statement 8`). The IM communicates to both vehicles the new (safe) velocities (`Statement 9`).

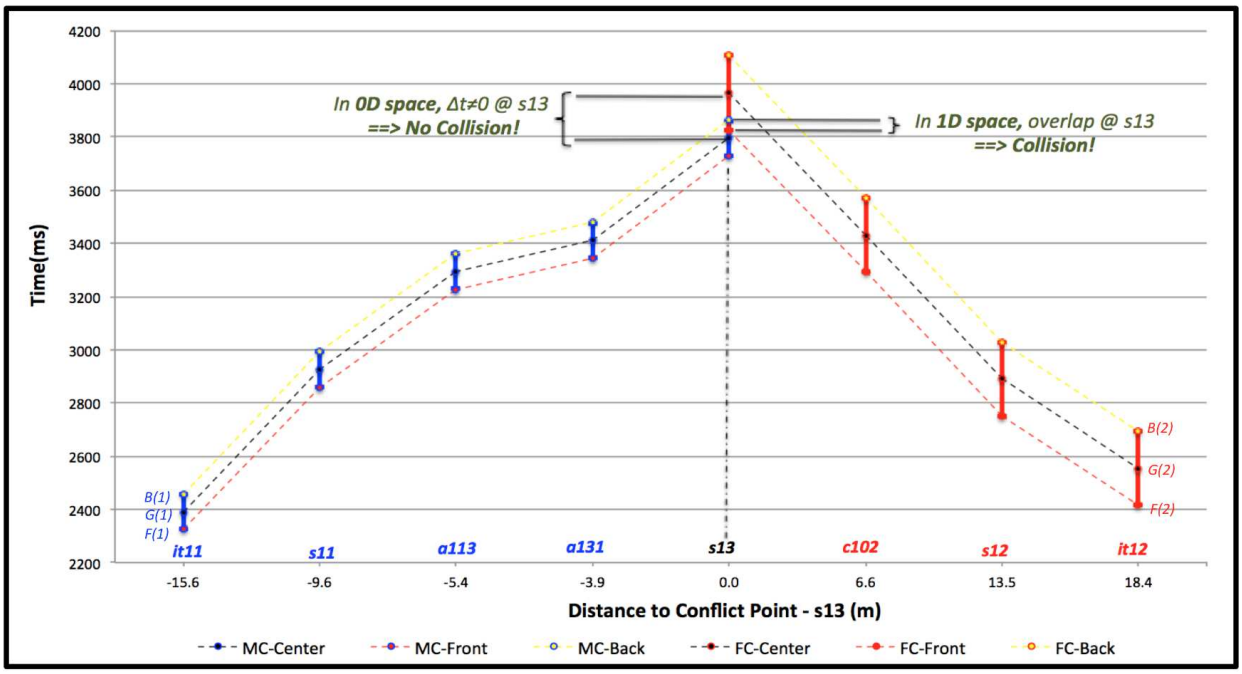


Figure 7.9: Space-time trajectory for two vehicles on a glancing collision course.

7.5.3 Impact of space ontological commitment on safety decisions

The results in the previous Section were obtained with $dimE = 0$ and $dimS = 1$ and, for the particular initial system state, the reasoner was able to predict, process and resolve the glancing collision before it occurred. However, it's not always the case as the shape and size of entities matter. To investigate this point in the context of this study, we run another simulation with $dimE = 0$ and $dimE = 1$. Instead of considering only the approach and collision points, we also include the prediction for intermediary, control points as pictured in the bottom right side of Figure 4.4. Using the physical model of the system and the capability of JavaFX, we run the simulation and extract both the distance to conflict point and time stamps at each location.

Figure 7.9 shows the resulting space-time trajectory of each vehicle. The distance of each control point to *s13* is computed and normalized with respect to the collision point as reference. In order for the reasoner to predict a collision, it needs the spatial data encapsulated in the geometric representation (spatial model) of the vehicle. When the vehicle geometry is a 2D point (0D space), its space-time trajectory is the black dashed line followed by the centroid of each vehicle, i.e., G(1) and G(2). The temporal gap between those two trajectories at *s13* indicates that the two vehicles (will) arrive at the collision point at different instances of time; in other words, no collision. However, if we consider a higher level spatial model (1D space) for the vehicle, i.e., a straight line by tracking its front and back as per Figure 4.3, we obtain two trajectories for each vehicle. The red one is the trajectory of the sensor installed at the front point F(i) of the vehicle while the yellow one is the one at the back B(i). The solid straight blue and red lines are the "temporal lengths" of vehicles 1 and 2, respectively. Now, we see that the back of

vehicle 1 arrives after the front of vehicle 2 and before its back. In other words, vehicle 1 gets to *s13* first and is hit on its right flank by vehicle 2. As such, *the 1D space model was able to predict a collision that the 0D space model was unable to catch !* Thus, the higher the space ontological commitment - characterized here by the value of $dimE$ - the more precise is the (safety) decision. However, the complexity of the reasoning increases too (in this example, the reasoner tracks 2 points in 1D representation instead of 1 in 0D). This results is consistent with the expectations as illustrated by the arrow in the right side of Figure 4.3.

Remark 7.6 *(Tolerance of temporal representations).* With Allen temporal calculus as the underlying theory (partially ordered domain) for temporal representations in models, the ontological commitment for time is clear in the CPS-KMoDS framework. However, the (system) modeler should be aware of, and is still responsible in defining the acceptable tolerance level for the temporal reasoning as done for space (see *Remark 7.2*). This is directly tied to the selected granularity of time in the model. It should be selected in a way that uncertainties in representations are minimized and kept in check so they don't affect decision making. As an illustration, if the *temporal tolerance* of the reasoning was set to Δ_t = 5s and the "Object" entity in `Statement 6` (see right side of Figure 7.8) was `Object : "110-303;120-300"` (in second), `Statement 7` would remain unchanged because the duration difference δ_t = 303-300=3s remains within the margin of error set by the value of Δ_t. However, with a lower temporal tolerance, let say Δ_t = 2s, `Statement 7` will return `"intDuring"` which, as per the representations in Figure 3.3, does not illustrate temporal conflicts at any extremity of the intervals.

Chapter 8: Conclusion and Future Work

8.1 Conclusions

8.1.1 Summary of Work

Our research conceptualizes and demonstrates formal procedures for a semantic modeling and reasoning framework that enables co-design of software (cyber) and hardware (physical), as well as integration of domain-specific semantics in model-based systems engineering (MBSE) of safety-critical cyber-physical systems (CPS). The latter are characterized by tight integration of software and physical processes, the need to satisfy stringent constraints on performance, safety and a reliance on automation for the management of system functionality. This work has been motivated by the realization of the inability of state-of-the-art traditional MBSE to support efficient design and verification of such systems. As demonstrated throughout this thesis, the need for highly-integrated system architectures in CPS changes the very nature of MBSE as currently perceived by scholars. Specifically, the separation of design concerns coupled with weak semantics of modeling languages (such as UML or SysML) lead to multiple distinctive viewpoints and a broken design flow which ultimately creates confusion and generates inconsistencies at every turn.

The work conducted focuses on a family of CPS applications for which safety and

performance are dependent on the correct predictions of future system state in terms of space and time. To assure system safety at all times, it is essential that these systems make the right decision at the right time and right place. This requirement, in turn, drives the need for collection and processing of temporal and spatial information in a timely manner. These observations have shaped our approach to "semantic-driven" design and analysis and, in particular, the need for models and strong semantics suitable for formal analysis. Prototype implementations employ Semantic Web technologies and demonstrate integration mechanisms among domains and meta-domains in Civil System applications, especially transportation systems.

8.1.2 Answers to Research Questions

Five research questions were asked and answered as follows:

1. How to effectively identify, capture and express safety requirements and physical semantics in the overall CPS design flow? Safety properties are hard constraints defined at system level thus, associated requirements define the criteria used to evaluate its operation. As such, safety requirements are non-functional and as shown in Chapter 5 can be formulated as decision problems with true/false or yes/no solutions. Physical semantics can be captured directly in the formal description of the physical subsystems (domain and metadomains) with linkage to the appropriate semantics as extensions (physical quantities for instance) as illustrated by the bottom two layers of the architecture in Figure 5.2. Also, in Chapter 7, we saw that, when translated into formal specifications supported by the corresponding requirements model (mapped to its semantics), requirements can be moved from their natural problem space to the solution space. This helps smooth the

development of safety Algorithms, streamlines the design flow and enables the co-design of both the physical and cyber parts of the system.

2. What temporal and space theories are the most appropriate for modeling and design of CPS? Based on foundation research that found interval-based models as most appropriate for formal analysis having time-dependent behavior and the selection criteria laid out, Allen's temporal interval calculus was identified as the most qualified temporal theory for our modeling framework. Similarly, the region connectedness calculus, which is a space-region theory was qualified as the most appropriate spatial formalism for the CPS design and modeling. Both theories were shown effective in supporting the compliance of resulting CPS models to requirements.

3. What knowledge representation formalism is suitable for semantic-based modeling and reasoning in CPS? Among the knowledge representation formalisms investigated, Description Logics offered very attractive features for semantics of complex domains (such as CPS) modeling including, non-finiteness of the domain and open-world assumption of the knowledge modeling. However, because of limitations on expressiveness of some DLs we identified selective extensions to the $\mathcal{ALC}$ resulting into the $\mathcal{SROIQ}$ DL. The latter provides support for concrete domains, improves the reasoning algorithms and ultimately supports the decidability of the knowledge modeling language OWL2.

4. To what extent can domain ontology models, especially the ones of time and space, and associated framework for formal reasoning about these meta-domains, be used to streamline design flows? We uncovered the "commonality of information" that crosses all domains and abstraction layers of CPS design. This allows ontology models to effectively encapsulate the relevant knowledge of the CPS of interest across application domains and

metadomains, especially space and time. The domain layer of the CPS-KMoDS in Figure 5.2 illustrates how the system domains can be encoded in semantic blocks. Given that requirements can be modeled in the similar way, this enables the flattening of the design knowledge and the bridging of the gap between the problem and the solution domains. Collision avoidance algorithms developed in Chapter 7 especially Algorithms 2 and 5, embody such integration.

5. How can cyber and physical behaviors be seamlessly integrated into an executable CPS model? The capabilities of the Whistle scripting language [65] currently under development were put to work to integrate the various modules of the CPS-KMoDS framework during the simulation. This has resulted into precise, accurate and executable CPS models appropriate for system level analyses. Also, even though the focus of this research was on semantic modeling, we've set and explored the foundations for the integration of the semantic and the physical models for the purpose of control as illustrated in applications in Chapter 4.

8.2 Future Work

The contributions of this thesis represent a proof-of-concept in the context of MBSE and, as such, there are many opportunities for extension and improvement. Looking forward, one goal is to support modeling of behaviors in large scale complex systems, such as cities and battlefields. An important opportunity is methodologies and tools that can provide assistance to cities in their recovery from environmental disruptions. Thus, a number of outstanding issues and problems are left for future research.

8.2.1 Ontological and Multi-level Integrated Control

In spatially distributed CPS, sending irrelevant control commands or correct ones with unacceptable delays as well as failing to provide commands at all when needed or making bad assumptions can lead to catastrophic outcomes. One such example is the recent accident involving a self-driving Google car which, leaving from a stopped position alongside a road, assumed it had the right of way over a bus driving on the targeted lane because it was ahead of it! This resulted in the vehicle glancing off the right side of the bus [300]. This makes the case for CPS controller to be intelligent i.e., its model and implementation have to be syntactically, semantically and ontologically correct [22]. Thus, the extension of the CPS-KMoDS with ontological control [79] will result into precise and accurate decision making at the control level, but also the ability for the control system to handle problematic control situations, especially those involving the violation of ontological assumptions.

On the other hand, state-of-the-art control for CPS often adopt a narrow view of the issue with focus on specific security aspects such as sensor data or weaknesses in operational implementation [43, 132] or various aspects of resiliency and robustness [281, 299]. When the view is broadened, domain semantics are barely taken into account across application domains including: cyber and physical systems control co-design with applications in distributed CPS [21], buildings [112] and, robotics [298]. There is value in investigating the synergistic integration of low and high level control with ontological control into a multi-level architecture. The resulting control scheme ought to be able to (1) guarantee precise and accurate handling of spatio-temporal constraints of the (Net)CPS of study, (2) make efficient decision based on sound reasoning and environment learning,

(3) resolve ontological assumption violations, with the ultimate goal of achieving system level safety.

8.2.2 Temporal and Spatial Reasoning with Uncertainties

Uncertainties are inherent to all domain knowledge representations as seen throughout this thesis, for space and time. Given the scope of this research our approach of this question has been limited to simple metrics for quantification of spatial and temporal tolerance as introduced in Chapter 7. More need to be done to tackle the double challenge of (i) representing uncertain and incomplete spatial and temporal knowledge and (ii) constructing effective methods of inference using those representations. For time, paths forward include graphical [173,278], fuzzy [73,190,293] and probabilistic representations [219,279]. Similarly, qualitative and quantitative approaches for dealing with uncertainties in spatial representations provide alternative investigation paths: fuzzy Sets [67,82,84,91,169], three-Valued Logics [52,159,202], rough sets [33,37], probabilistic [167,250,269] and hybrid Methods [244,245].

8.2.3 Whistle Platform Development

To date, Whistle [65] has been developed with engineering analysis of multi-domain applications in mind. Despite its current impressive features, many pieces of a comprehensive systems integration engine are missing. Opportunities for future work include language support for the smooth integration of models of physical components and computation along with the simulation and visualization of continuous and discrete behaviors. Whistle library support for calendar, scheduling and optimization modules is needed.

This will enable tasks planning and scheduling, which would allow the implementation, simulation and design analysis of increasingly realistic, real-world scenarios. Another need is computational support for elicitation of requirements, followed by their integration into the design flow. Such a capability would open the door to their (automatic) verification as part of the design process. The latter is the mechanism through which the solution domain (design) is checked against the problem domain (requirements) and should be carefully defined and configured. Also, these capabilities will enable the simulation and analysis of large scale complex systems.

8.2.4 Safe Airport Taxiway System

A key problem with aviation systems becoming progressively crowded is their diminished ability to deal with heavy work loads and enhanced ground safety concerns [273]. Since 1990, the Federal Aviation Administration (FAA) has reported six (6) runway collisions resulting in sixty-three (63) deaths. Fifty-four (54) percent of *incursions* during 2003 through 2006 were caused by pilot errors and (29) percent were caused by air traffic controller errors [272, 273]. Also, state-of-the-art solutions have been unable to prevent the frequent occurrence of aircraft wings and tails clipping on airport taxiways all around the world [14, 138, 164]. Recent work [50, 88, 145, 146] in the use of tree structures for the hierarchical organization of temporal and spatial refinements offers one potentially useful pathway forward.

We envision solutions that will rely on autonomous agents that implement semantic-based, spatio-temporal collision avoidance algorithms introduced in this dissertation. Figure 8.1 shows the current state of this research with a portion of the Baltimore-Washington

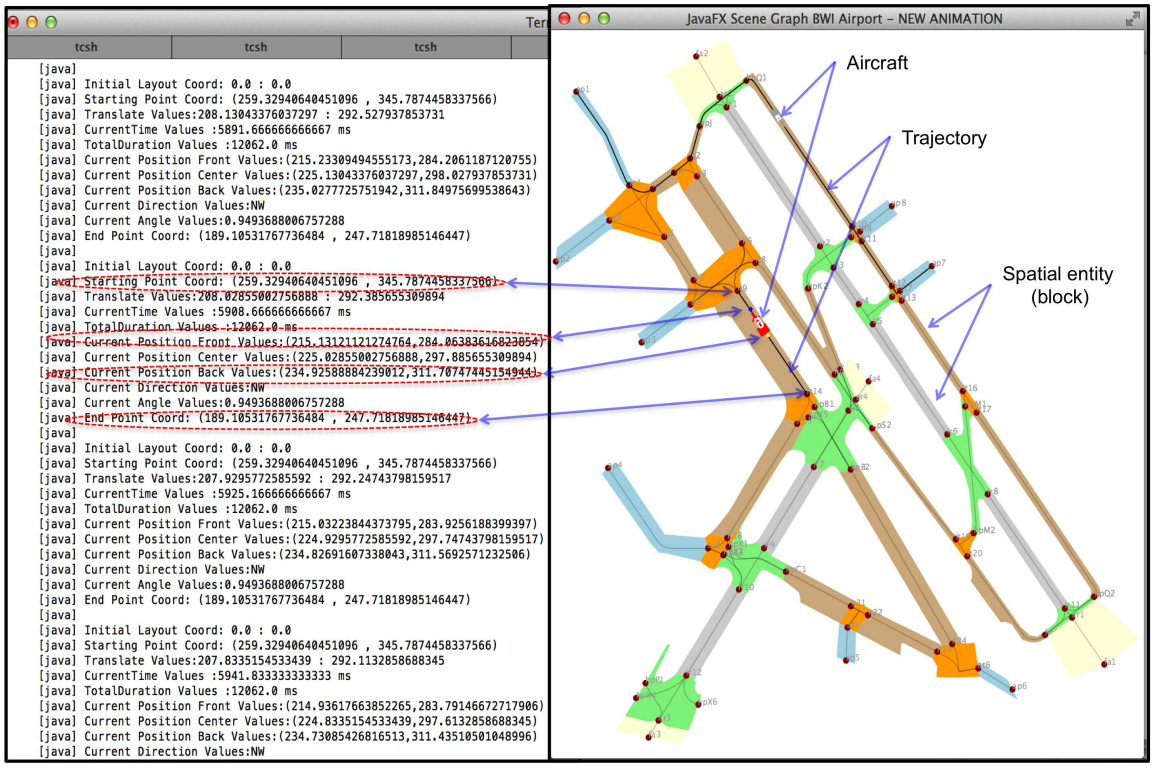


Figure 8.1: Real-time simulation and safety validation of taxiway operations

International (BWI) airport. So far, the focus of the work has been on developing the physical model of the system. The right hand side of the figure represents Runway 15L/33R and surrounding runway and taxiways. The map is created using parsed data from open street map and visualized with the help of JavaFX. Runways and taxiways are partitioned into spatial entities consistent with the framework in Figure 4.3. Aircraft are also created as spatial objects and assigned predefined trajectories (black lines). Behaviors are defined for each type of spatial entity and executed by the aircraft. On the left side of the figure, the position of the aircraft is tracked in real time using sensors located at specific areas (front and back in this case). The feasibility of implementation in real-time of solutions, such as the ones described here, requires studies that incorporate delays and the effect they will have on achievable levels of performance.

When complete, this research will produce new approaches to the multi-level simulation of concurrent physical behaviors, coupled with mapping relationships and software tools for the reasoning, evaluation, and coordination of taxiway operations defined in terms of their spatial and temporal requirements. Also, this application offers an ideal platform for in-depth investigation of the issues identified in Sections 8.2.1 through 8.2.3.

Appendix A: Description Logics and $\mathcal{ALC}$ Extension

A.1 Basic description logics

Description logics are a family of logic-based knowledge representation formalisms that can describe domain in terms of concepts (e.g., classes in OWL), roles (e.g., properties, relationships) and individuals (e.g., objects). As a subset of first-order logics (FOL), they provide well-defined semantics supporting decidability and development of efficient reasoning algorithms. The acronym $\mathcal{AL}$ stands for attribute language (see Appendix 1 of Baader [29] for details on naming scheme for DLs). When a basic DL serves as a foundation for knowledge representation, many other DLs may be constructed through the addition of specific extensions. One such extension is the attribute language concepts ($\mathcal{ALC}$). The benefit of this extension mechanism is that is allows for the specification of languages supporting new features. For example, atomic concepts (A) can be extended to support arbitrary concepts (C), thereby enabling the description of any domain of interest. A second important extension is the number restriction N which leads to ALCN DL." This is a subset of the frame-based DL $\mathcal{FL}$ and is equivalent to $\mathcal{AL}$, but without atomic negation, inverse, transitive roles and subroles or concrete domains [**?**, **?**]. As we will soon see, these extensions and restrictions are needed to make the language decidable with low complexity, a strategy that is supported by Lutz [172], who identifies $\mathcal{ALC}$ as the most appropriate DL for reasoning with concrete domains.

A.2 The $\mathcal{ALC}$ description logics

In this DL, the operators universal ($\forall$), existential ($\exists$), intersection ($\sqcap$), Union ($\sqcup$), negation ($\neg$) can be properly applied to atomic (A, A_i,..), arbitrary (C, D,..), top ($\top$) (i.e., All concepts names) and bottom($\bot$) (i.e., Empty concept) concepts. Primitive relations (r, s,...) as well as existential restriction ($\exists r.C$) and value restriction ($\forall r.C$) on concepts are other key constructors used to formally define a domain of interest. The complete set of defined concepts of the basic $\mathcal{ALC}$ system can be represented by the following grammar:

C:= $\top \mid \bot \mid A \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \forall r.C \mid \exists r.C$

For instance, the statement *"A woman who is single and whose children are either boy or girl"* can be expressed in DL using a minimal number of concepts as follows.

Human $\sqcap$ $\neg$ Male $\sqcap$ $\forall$hasChild.(Boy $\sqcup$ Girl).

In DL, semantics are defined by interpretations. In the case of $\mathcal{ALC}$, an interpretation $\mathcal{I}$ is formally defined as follows [26]:

Definition 1 *(Interpretation)*: An interpretation $\mathcal{I} = (\triangle^{\mathcal{I}}, .^{\mathcal{I}})$ consists of a non-empty set $\triangle^{\mathcal{I}}$, called the domain of $\mathcal{I}$, and a function $.^{\mathcal{I}}$ that maps every $\mathcal{ALC} - concept$ to a subset of $\triangle^{\mathcal{I}}$, and every role name to a subset of $\triangle^{\mathcal{I}} \times \triangle^{\mathcal{I}}$ such that, for all $\mathcal{ALC} - concepts$ C, D and all role names r,

$\top^{\mathcal{I}} = \triangle^{\mathcal{I}}$, $\bot^{\mathcal{I}} = \emptyset$

$(C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$, $(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$, $\neg C^{\mathcal{I}} = \triangle^{\mathcal{I}} \setminus C^{\mathcal{I}}$,

$(\exists r.C)^{\mathcal{I}} = \{x \in \triangle^{\mathcal{I}} \mid \exists y \in \triangle^{\mathcal{I}} with < x, y > \in r^{\mathcal{I}} and \;\; y \in C^{\mathcal{I}}\}$

$(\forall r.C)^{\mathcal{I}} = \{x \in \triangle^{\mathcal{I}} \mid \forall y \in \triangle^{\mathcal{I}}, if < x, y > \in r^{\mathcal{I}}, \;\; then \;\; y \in C^{\mathcal{I}}\}$

x and y are *instances* of C in the interpretation $\mathcal{I}$.

Concept descriptions are used to build statements in a DL knowledge base, in accordance to the semantics provided by the interpretation.

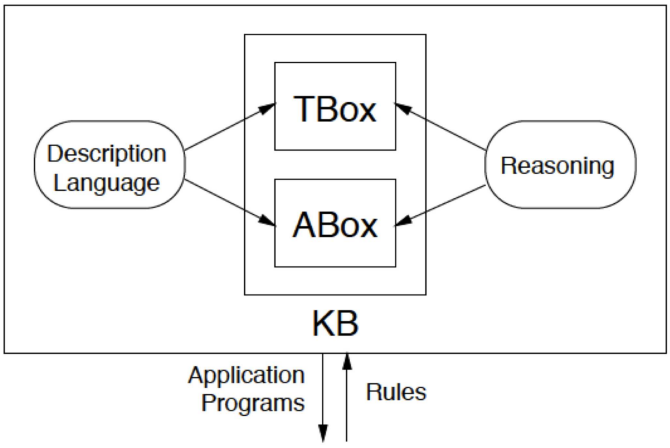


Figure A.1: High level architecture of a knowledge representation system based on description logics (Source: [29])

Figure A.1 shows that the knowledge base (KB) is typically made up of two parts: (1) A terminological part or TBox, and (2) An assertional part called ABox. TBox statements describe a set of concepts and properties for these concepts. ABox statements are TBox-compliant statements about individuals belonging to those concepts. Together ABox and TBox statements make up a knowledge base.

Definition 2 *(TBox)*: A TBox $\mathcal{T}$ is a finite set of general concept inclusion (GCI). A GCI is of the form $C \sqsubseteq D$ where C, D are $\mathcal{ALC} - concepts$. When $C \equiv D$ the corresponding pair of GCI $C \sqsubseteq D$ and $D \sqsubseteq C$ are symmetrical. If C is a concept name, then the axiom $C \equiv D$ is called a *definition*. An interpretation $\mathcal{I}$ is a model of a GCI $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$; $\mathcal{I}$ is a model of a TBox $\mathcal{T}$ if it is a model of every CGI in $\mathcal{T}$.

A TBox $\mathcal{T}$ can be definitorial (also called an acyclic TBox), i.e., it contains only definitions along with certain restrictions. In this case, concept names in left-hand side of $\mathcal{T}$ are

Name	Syntax	Semantics	Symbol
Top	$\top$	$\Delta^{\mathcal{I}}$	$\mathcal{AL}$
Bottom	$\bot$	$\emptyset$	$\mathcal{AL}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$	$\mathcal{AL}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$	$\mathcal{U}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}$	$\mathcal{C}$
Value restriction	$\forall R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.\ (a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$	$\mathcal{AL}$
Existential quant.	$\exists R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b.\ (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$	$\mathcal{E}$
Unqualified number restriction	$\geqslant n\, R$ $\leqslant n\, R$ $=n\, R$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\rvert \geq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\rvert \leq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}\rvert = n\}$	$\mathcal{N}$
Qualified number restriction	$\geqslant n\, R.C$ $\leqslant n\, R.C$ $=n\, R.C$	$\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \geq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert \leq n\}$ $\{a \in \Delta^{\mathcal{I}} \mid \lvert\{b \in \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\rvert = n\}$	$\mathcal{Q}$
Role-value-map	$R \subseteq S$ $R = S$	$\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow (a,b) \in S^{\mathcal{I}}\}$ $\{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \leftrightarrow (a,b) \in S^{\mathcal{I}}\}$	
Agreement and disagreement	$u_1 \doteq u_2$ $u_1 \neq u_2$	$\{a \in \Delta^{\mathcal{I}} \mid \exists b \in \Delta^{\mathcal{I}}.\ u_1^{\mathcal{I}}(a) = b = u_2^{\mathcal{I}}(a)\}$ $\{a \in \Delta^{\mathcal{I}} \mid \exists b_1, b_2 \in \Delta^{\mathcal{I}}.\ u_1^{\mathcal{I}}(a) = b_1 \neq b_2 = u_2^{\mathcal{I}}(a)\}$	$\mathcal{F}$
Nominal	I	$I^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ with $\lvert I^{\mathcal{I}}\rvert = 1$	$\mathcal{O}$

Figure A.2: Summary of description logic concepts constructors ([29]).

"defined concepts" while concepts in the other side are "primitive" concepts."

Definition 3 *(ABox)*: An ABox $\mathcal{A}$ is a finite set of assertional axioms of the form x : C or (x,y) : r, where C is an $\mathcal{ALC} - concept$, r is an $\mathcal{ALC} - role$, and x and y are individual names. An interpretation $\mathcal{I}$ is a model of an assertional axiom x : C if $x^{\mathcal{I}} \in C^{\mathcal{I}}$ and $\mathcal{I}$ is a model of an assertional axiom (x,y): r if $(x^{\mathcal{I}}, y^{\mathcal{I}}) \in r^{\mathcal{I}}$; $\mathcal{I}$ is a model of an ABox $\mathcal{A}$ if it is a model of every axiom in $\mathcal{A}$.

These definitions equip us with the necessary elements to formally define the notion of knowledge base introduced above.

Definition 4 *(Knowledge base)*: A knowledge base (KB) is a pair $(\mathcal{T}, \mathcal{A})$ where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is an ABox. An interpretation $\mathcal{I}$ is a model of a KB $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ if $\mathcal{I}$ is an interpretation of $\mathcal{T}$ and $\mathcal{I}$ is a model of $\mathcal{A}$.

We write $\mathcal{I} \models \mathcal{K}$, ($\mathcal{I} \models \mathcal{T}$, $\mathcal{I} \models \mathcal{A}$) to denote that $\mathcal{I}$ is a model of a KB $\mathcal{K}$ (respectively, TBox $\mathcal{T}$, ABox $\mathcal{A}$).

A summary of the main DL concept constructors is shown in Figure A.2.

Appendix B: DL extensions for OWL2

1. **Role hierarchy ($\mathcal{H}$):** Hierarchies between roles are allowed in this extension. This results into the $\mathcal{ALCH}$ or $\mathcal{SH}$ DL formalism that is a translation of foundational OWL. In turn, there are three OWL sublanguages with increasing expressiveness: OWL-Lite, OWL-DL and OWL-Full(no syntactic constraints). More precisely, the DL TBox along with the role hierarchy extension map to the OWL (Lite or DL) ontology. In OWL, the domain of interest is defined in term of *classes* related to each other by *properties*. These entities correspond respectively to concepts and roles in $\mathcal{SH}$ DL.

   As an illustration, the DL statement $hasColor.CarColor \sqsubseteq hasCarDescriptor$ can be translated into OWL as follows:

```
<owl:ObjectProperty rdf:about="#hasColor">
  <rdfs:subPropertyOf rdf:resource="#hasCarDescriptor"/>
  <rdfs:range rdf:resource="#CarColor"/>
</owl:ObjectProperty>
```

   The properties here are of type object, but they could also be of type data depending on the domain and application need.

2. **Nominal ($\mathcal{O}$):** In this DL extension, use of the nominal constructor {} allows for the definition of singleton sets (i.e., as concepts) from individual names. The

corresponding restriction in OWL is achieved with the object property elements *owl:oneOf* and *owl:hasValue*.

Let us suppose that we are given an "individual" V6. We can use this extension to define all cars that are equipped with this particular engine type as follows.

$Car \sqcap \exists hasEngine.\{V6\}$

This can be translated in OWL using the constructor *owl:hasValue* as follows.

```
<owl:Class rdf::about="#Engine">
 <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasEngine"/>
    <owl:hasValue   rdf:resource="#V6"/>
   </owl:Restriction>
 </rdfs:subClassOf>
</owl:Class>
```

An important limitation of nominals [26] is that is can dramatically increase the complexity of reasoning processes.

3. **Inverse and transitive roles($\mathcal{I}$):** This extension is needed to increase the expressiveness of the DL. Inverse and transitive roles are expressed in OWL using the object properties *owl:inverseOf* and *owl:TransitiveProperty*.

   For instance, $makeCar \equiv hasMaker^{-}$ and can be expressed in OWL as follows.

```
<owl:ObjectProperty rdf:about="#makeCar">
  <owl:inverseOf rdf:resource="#hasMaker"/>
</owl:ObjectProperty>
```

   The procedure for expressing that an object property is transitive is as follows:

```
<owl:ObjectProperty rdf:about="#hasFollower">
  <rdf:type rdf:resource="#TransitiveProperty"/>
  <rdfs:domain rdf:resource="#Car"/>
  <rdfs:range rdf:resource="#Car"/>
</owl:ObjectProperty>
```

4. **Cardinality/number restriction ($\mathcal{N}$):** This extension allows for the formal expression of the number of relationships that individuals of specific types can have among them, a feature that is particularly relevant to CPS modeling.

   For example, the statement *A car has at most one engine* can be written as follows.

   $Car \sqsubseteq \leq 1 hasEngine$

   Additional syntax elements and their corresponding semantics are shown on Figure A.2.

5. **Qualified number restrictions ($\mathcal{Q}$):** This extension is similar to the previous one with the difference that we can describe individual types that are counted by a given number of expressions, which allows for representation of the notion of a "data interval."

   To see how this works in practice, we can extend the definition of a car to allow for two through five doors. The corresponding logical expression is:

   $Car \equiv Vehicle \sqcap \leq 1 hasEngine \sqcap (\geq 2 hasDoor \sqcap \leq 5 hasDoor)$

6. **Role restrictions ($\mathcal{R}$):** This extension completes the $\mathcal{I}$ DL by providing role inclusion axioms as well as support for reflexivity, symmetry and roles disjointness. In OWL, these features show up as the property characteristics *owl:reflexive*,

owl:irreflexive, *owl:symmetry*, *owl:functional* and *owl:disjointWith*. The fragment of code:

```
<owl:ObjectProperty rdf:about="#hasFollower">
  <rdf:type rdf:resource="#TransitiveProperty"/>
  <rdf:type rdf:resource="#IrreflexiveProperty"/>
  <rdf:type rdf:resource="#AsymmetricProperty"/>
  <owl:disjointWith rdf:resource="#hasPredecessor"/>
  <rdfs:domain rdf:resource="#Car"/>
  <rdfs:range rdf:resource="#Car"/>
</owl:ObjectProperty>
```

illustrates the use of these characteristics for a more precise specification of the aforementioned `hasFollower` object property.

7. **Concrete domains:** This extension provides support for the handling of *concrete sets* (real numbers, integers, strings, etc..) and *concrete predicates* (numerical comparisons, string comparisons and comparisons with constants) on these sets.

Appendix C: Reasoning services for $\mathcal{SROIQ}$ - based ontologies

The $\mathcal{SROIQ}$ description logics that support OWL 2 are introduced in Krotzsch [143] and are thoroughly detailed in Horrocks [128]. Here, $\mathcal{SR}$ = $\mathcal{ALC}$ + role chains($\mathcal{R}$), $\mathcal{O}$ = nominals (closed classes), $\mathcal{I}$ = support for inverse rules, and $\mathcal{Q}$ = qualified cardinality restrictions. We note the extension of the grammar to include *role expressions* $\mathbf{R} := \mathbf{U} \mid \mathbf{N_R} \mid \mathbf{N_R^-}$ where $\mathbf{N_R}$ is the set of role names and $\mathbf{U}$ is the universal role. Also, alongside the TBox and ABox, the *RBox* is an integral part of $\mathcal{SROIQ}$ axioms.

Thus, from the grammar

C:= $N_C \mid \mathbf{C} \sqcap \mathbf{C} \mid \mathbf{C} \sqcup \mathbf{C} \mid \neg\mathbf{C} \mid \top \mid \bot \mid \forall\mathbf{R}.\mathbf{C} \mid \exists\mathbf{R}.\mathbf{C} \mid \geq n\mathbf{R}.\mathbf{C} \mid \leq n\mathbf{R}.\mathbf{C} \mid \exists\mathbf{R}.Self \mid \{N_I\}$

where n is a non-negative integer, $\mathbf{C} \sqcap \mathbf{C}$ representing expressions of the form $C \sqcap D$ with C, D $\in \mathbf{C}$ and $\{N_I\}$ are individual names, $\mathcal{SROIQ}$ axioms are defined as follows.

ABox: $\mathbf{C}(N_I)$ $\mathbf{R}(N_I, N_I)$ $N_I \approx N_I$ $N_I \not\approx N_I$

TBox: $\mathbf{C} \sqsubseteq \mathbf{C}$ $\mathbf{C} \equiv \mathbf{C}$

RBox: $\mathbf{R} \sqsubseteq \mathbf{R}$ $\mathbf{R} \equiv \mathbf{R}$ $\mathbf{R} \circ \mathbf{R} \sqsubseteq \mathbf{R}$ *Disjoint*(**R**,**R**)

When applied to any given $\mathcal{SROIQ}$-based ontology, this set of axioms creates a knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$ for the domain being modeled. However, when applicable, the following reasoning tasks are required with regard to the TBox $\mathcal{T}$ and ABox $\mathcal{A}$ [196]:

Definition C.0.1. *(Satisfiability)* A concept C is satisfiable *w.r.t. a TBox* $\mathcal{T}$ if there exists an interpretation $\mathcal{I} \models \mathcal{T}$ such that $C^I \neq \emptyset$.

Similarly, an ABox $\mathcal{A}$ is satisfiable *w.r.t. a TBox* $\mathcal{T}$ if there exists an interpretation $\mathcal{I} \models \mathcal{T} \cup \mathcal{A}$.

Definition C.0.2. *(Subsumption)* A concept C is subsumed by D ($C \sqsubseteq_{\mathcal{T}} D$) with C, D $\in$ **C** if for all interpretations $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$ then $C^I \subseteq D^I$.

Definition C.0.3. *(Equivalence)* Two concepts C and D (C, D $\in$ **C**) are equivalent with respect to $\mathcal{T}$ if for all interpretations $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$ then $C^I = D^I$

Definition C.0.4. *(Disjointness)* Two concepts C and D (C, D $\in$ **C**) are disjoint with respect to $\mathcal{T}$ if for all interpretations $\mathcal{I}$, if $\mathcal{I} \models \mathcal{T}$ then $C^I \cap D^I = \emptyset$

The reasoner should be able to systematically decide on the existence and satisfaction of these characteristics and assert (or infer) new facts and statements that are added to the knowledge base $\mathcal{K}$. However, reasoning over $\mathcal{K}$ in its wholeness is very inefficient. Fortunately, it has been proven that there are ways to reduce the complexity of reasoning to polynomial order through elimination of ABox and TBox axioms/concepts. This advance is formulated in the following results [128, 196]:

Proposition 1.1. *(Satisfiability w.r.t. TBox)* Subsumption, equivalence, and disjointness with respect to $\mathcal{T}$ are reducible to testing (un)satisfiability w.r.t. $\mathcal{T}$

Lemma 2. *(ABox Elimination)* $\mathcal{SROIQ}$ concept satisfiability with respect to ABoxes, RBoxes, and TBoxes is polynomially reducible to $\mathcal{SROIQ}$ concept satisfiability with respect to RBoxes and TBoxes only.

Similar result is formulated for the elimination of both the TBox and Universal Role thus, the following theorem addressing reduction.

Theorem 3 (Reduction)**.** *1. Satisfiability and subsumption of* $\mathcal{SROIQ}$*-concepts w.r.t. ABoxes, RBoxes, and TBoxes are polynomially reducible to (un)satisfiability of* $\mathcal{SROIQ}$*-concepts w.r.t. RBoxes.*

2. Without lost of generality, we can assume that RBoxes do not contain role assertions of the form Irr(R), Tra(R), or Sym(R), and that the universal role is not used.

This result reduces the standard $\mathcal{SROIQ}$ (concepts and ABoxes) inference problem to the one of determining the consistency of a $\mathcal{SROIQ}$-concept with respect to a reduced RBox where all role assertions are of the form *Ref(R)* or *Dis(R,S)*. Krotzsch [143] also points out the need for "structural restrictions" on $\mathcal{SROIQ}$-based ontologies as a whole in order to guarantee the existence of correct and terminating algorithms to support inferencing. We note that the first restriction, simplicity is concerned with non-simple roles resulting from roles composition. Second, regularity is concerned with RBox axioms. The main goal of such restrictions is to limit the occurrence of cyclic dependencies between complex roles and inclusion axioms (i.e., see the OWL constructor owl:SuperPropertyOf (chain)). Horrocks et al. [128] build on these results to develop and describe a terminating, sound, and complete tableau-based algorithm that decides the consistency of a $\mathcal{SROIQ}$-concepts with respect to a reduced RBox.

Theorem 4 (Decidability)**.** *The tableau algorithm decides satisfiability and subsumption of $\mathcal{SROIQ}$-concepts with respect to ABoxes, RBoxes, and TBoxes.*

Appendix D: Multi-dimensional Spatial Representation Functions for Safety-Critical CPS Design

D.1 Assumptions and Foundations

The CPS of interest is made of static and dynamic components that interact with each other. Each group of subsystem is made of different types of components. Components of same types share the same type of behavior. The tree structure on Figure D.1 shows a high level description of the structural decomposition of the system. As an illustration, for a traffic intersection system, traffic lights and radars are static components while (motor) vehicles and pedestrians are dynamic components.

The number of static and dynamic components in the system are derived from the number of components of each type as follows.

$$n_{SC} = \sum_{i=0}^{s-1} n_{TSC}[i] \tag{D.1}$$

$$n_{DC} = \sum_{i=0}^{d-1} n_{TDC}[i] \tag{D.2}$$

The total number of components in the system results from the sum of the two variables above.

$$n_C = n_{SC} + n_{DC} \tag{D.3}$$

The system evolves in a connected world in which components can be connected. Thus, each component is smart i.e. it can possess the following 3 capabilities: (a) communication with cooperating objects within a certain sensing range, (b) sensing and detection for safety, based on the physics of the component such as its current speed or some design prescription such as a security perimeter around the component, and (c) boundaries sensing for physical contact detection.

The egg-shape graphics on the left hand side of figure D.2 illustrate the spatial representation of these capabilities for each component $C[i]$. The biggest oval $S_{cov}[i]$ represents the coverage space for communication purpose while $S_{phys}[i]$) and $S_{cont}[i]$ represent the physics and contact detection spaces. All these entities are centered on the centroid point of the spatial representation of the component which is assumed to be the one of the shape representing the object. Equipped with these elements, we make the following assumptions regarding the spatial representations for the various capabilities of the components. We write $I = \{0, ..., n\}$ the set of component identifiers in the system.

A1 - Each component occupies some spatial resource within the boundaries of the system. This translates as follows:

$$\forall i \in I, S_{cont}[i] \neq \emptyset \tag{D.4}$$

A2 - When the physical (communication) space exists, it contains the contact

(physical) space. This translates as follows.

$$\forall i \in I, S_{phys}[i] \neq \emptyset \Rightarrow S_{cont}[i] \subset S_{phys}[i] \tag{D.5}$$

$$\forall i \in I, (S_{cov}[i] \neq \emptyset \wedge S_{phys}[i] \neq \emptyset) \Rightarrow S_{phys}[i] \subset S_{cov}[i] \tag{D.6}$$

The explanation behind (D.5) is that each smart component, in its normal mode of operation, has the ability to detect and sense components that are within its vicinity as pertaining to its current dynamics. As for (D.6) the rationale is that the coverage and communication range of a component exceeds the one of the constrained by its physics. In open space, technology such as GPS allow almost unlimited coverage to moving object on earth and in the air.

D.2 Interaction Functions

Interaction functions define and describe interactions between pair of component in the system in a formal and systematic way. The table on figure D.3 shows the mapping of spatial entities for the purpose of the definition of interaction functions between components i.e. f^p_{com} for communication, f^p_{saf} for safety and f^p_{crash} for contact capabilities. For each capability, we define a function based on the intersection of the tree types of spatial entities as follows.

The right hand side of Figure D.2 shows the values taken by the functions defined in (D.7),(D.8) and (D.9) for different configurations of the system. For simplification, we use the following notations $com(i,j)$, $saf(i,j)$ and $crash(i,j)$ respectively for $f^p_{com}(i,j)$, $f^p_{saf}(i,j)$ and $f^p_{crash}(i,j)$.

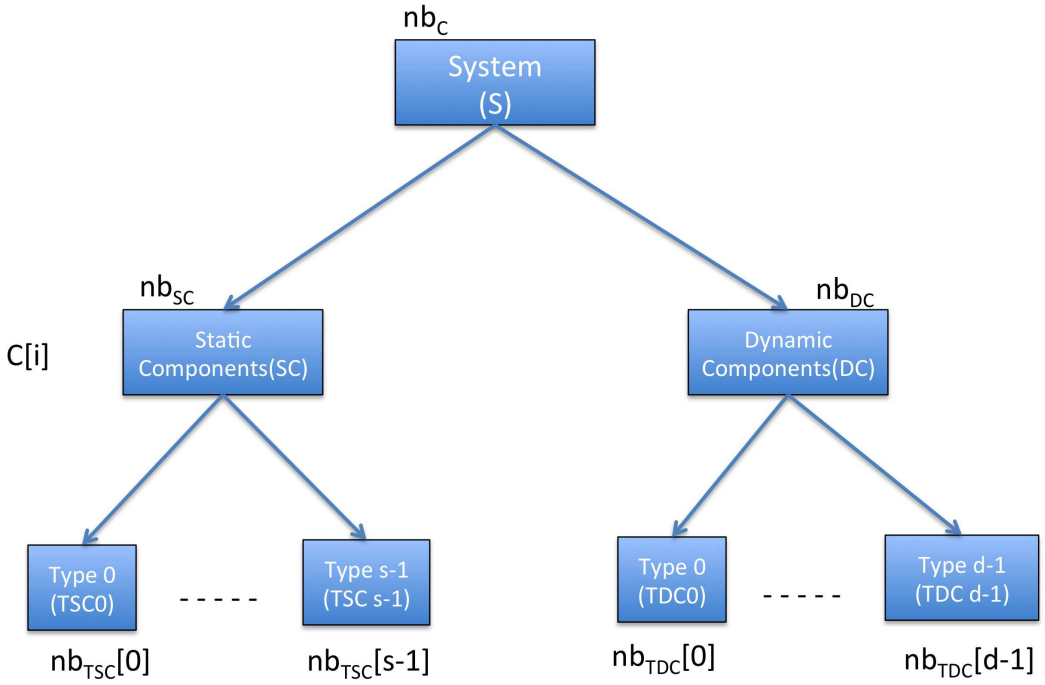


Figure D.1: Structural decomposition of the system.

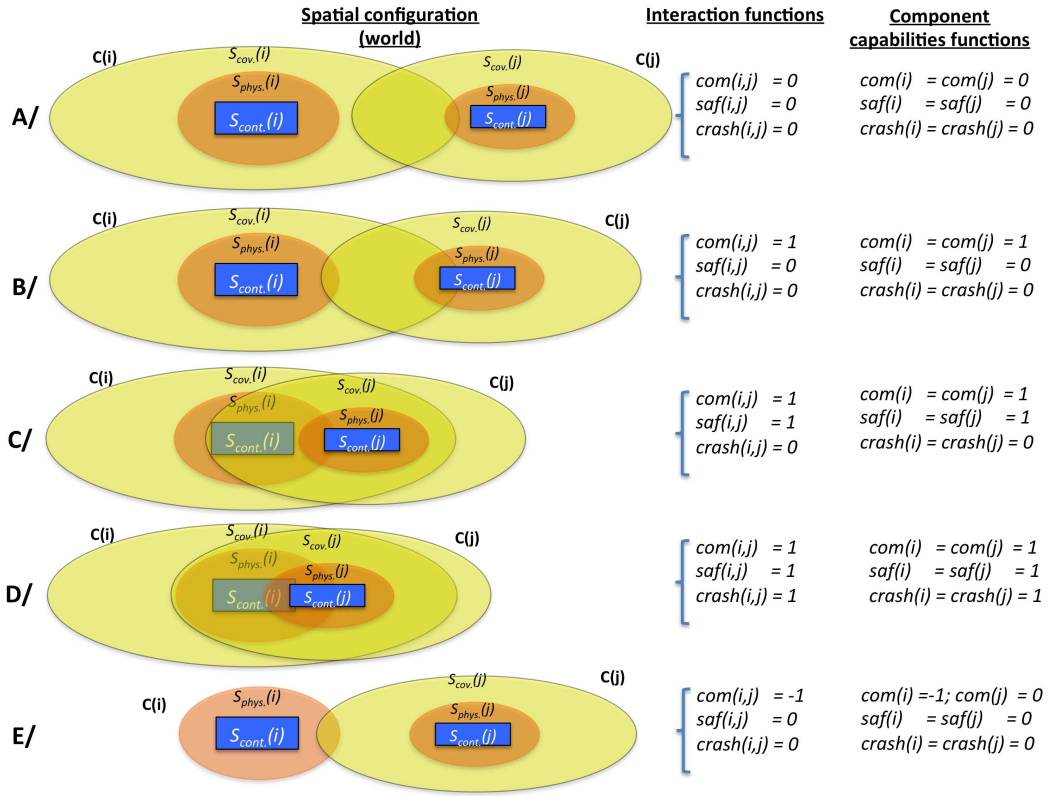


Figure D.2: Spatial representation and corresponding spatial functions.

Intersection(*connects, overlaps, parts, identical*)		RCC-compliant Spatial entity(j)		
		Coverage- $S_{cov.}[j]$	Physics – $S_{phys.}[j]$	Contact – $S_{cont.}[j]$
RCC-compliant Spatial entity(i)	Coverage - $S_{cov.}[i]$	*NA*	*NA*	$f^p_{com.}(i,j)$
	Physics – $S_{phys.}[i]$	*NA*	*NA*	$f^p_{saf.}(i,j)$
	Contact – $S_{cont.}[i]$	$f^p_{com.}(i,j)$	$f^p_{saf.}(i,j)$	$f^p_{crash.}(i,j)$

Figure D.3: Interaction Functions Definition

$$f^p_{com}(i,j) = f^p_{com}(j,i) = \begin{cases} -1 & \text{if } S_{cov}[i] = \emptyset \vee S_{cont}[j] = \emptyset \\ 0 & \text{if } S_{cov}[i] \cap S_{cont}[j] = \emptyset \vee S_{cov}[j] \cap S_{cont}[i] = \emptyset \vee i = j; \\ 1 & \text{if } S_{cov}[i] \cap S_{cont}[j] \neq \emptyset \vee S_{cov}[j] \cap S_{cont}[i] \neq \emptyset; \end{cases} \tag{D.7}$$

$$with \quad S_{cov}[i], S_{cov}[j] \neq \emptyset \text{ when } f^p_{com}(i,j) = 0, 1 \ .$$

$$f^p_{saf}(i,j) = f^p_{saf}(j,i) = \begin{cases} -1 & \text{if } S_{phys}[i] = \emptyset \\ 0 & \text{if } S_{phys}[i] \cap S_{cont}[j] = \emptyset \vee S_{phys}[j] \cap S_{cont}[i] = \emptyset \vee i = j; \\ 1 & \text{if } S_{phys}[i] \cap S_{cont}[j] \neq \emptyset \vee S_{phys}[j] \cap S_{cont}[i] \neq \emptyset; \end{cases} \tag{D.8}$$

$$with \quad S_{phys}[i], S_{phys}[j] \neq \emptyset \text{ when } f^p_{saf}(i,j) = 0, 1.$$

$$f^p_{crash}(i,j) = f^p_{crash}(j,i) = \begin{cases} 0 & \text{if } S_{cont}[i] \cap S_{cont}[j] = \emptyset \vee S_{cont}[j] \cap S_{cont}[i] = \emptyset \vee i = j \\ 1 & \text{if } S_{cont}[i] \cap S_{cont}[j] \neq \emptyset \vee S_{cont}[j] \cap S_{cont}[i] \neq \emptyset \end{cases} \tag{D.9}$$

Table D.1: Definition of communication, safety and crash interaction functions

D.3 Component Capability Functions

We rely on the interaction functions above to define for a given component $C[i]$, its communication($f^c_{com}(i)$), safety($f^c_{saf}(i)$) and crash ($f^c_{crash}(i)$) functions as follows.

Communication Capability.

$$f^c_{com}(i) = \begin{cases} \sum_{j=0}^{n_C-1} \alpha^{com}_j * f^p_{com}(i,j) & \text{if } S_{cov}[i] \neq \emptyset \\ -1 & \text{if } S_{cov}[i] = \emptyset \end{cases} \tag{D.10}$$

The parameter α^{com}_j in (D.10) is defined as follows .

$$\alpha^{com}_j = \begin{cases} 1 & \text{if } f^p_{com}(i,j) = 1 \\ 0 & \text{if } f^p_{com}(i,j) \neq 1 \end{cases} \tag{D.11}$$

Safety Capability.

$$f^c_{saf}(i) = \begin{cases} \sum_{j=0}^{n_C-1} \alpha^{saf}_j * f_{saf(i,j)} & \text{if } S_{phys}[i] \neq \emptyset \\ -1 & \text{if } S_{phys}[i] = \emptyset \end{cases} \tag{D.12}$$

The parameter α^{saf}_j in (D.12) is defined as follows .

$$\alpha^{saf}_j = \begin{cases} 1 & \text{if } f^p_{saf}(i,j) = 1 \\ 0 & \text{if } f^p_{saf}(i,j) \neq 1 \end{cases} \tag{D.13}$$

Crash Capability.

$$f^c_{crash}(i) = \sum_{j=0}^{n_C-1} f^p_{crash}(i,j) \tag{D.14}$$

Bibliography

[1] 2004. *W3*; See https://www.w3.org/TR/owl-features/, February 2004.

[2] 2006. *Time Ontology in OWL*, accessible at : http://www.w3.org/TR/owl-time/; Accessed 11/24/2013.

[3] 2011. *At a crossroads:New research predicts which cars are likeliest to run lights at intersections.*; Accessible at : http://newsoffice.mit.edu/2011/driving-algorithm-1130; last accessed on 06/05/2014.

[4] 2013. *Apache Jena*, accessible at : http://www.jena.apache.org; Accessed on 11/27/13.

[5] 2013. *Jscience*, accessible at : http://www.jscience.org; Accessed on 11/27/13.

[6] 2014. *Suggested Upper Merged Ontology (SUMO)*, Available at http://http://www.ontologyportal.org/, Accessed on 05/27/14.

[7] 2014. *OpenCyc*, Available at http://www.cyc.com/platform/opencyc, Accessed on 06/01/14.

[8] 2016. *RACER*; See http://www.ifis.uni-luebeck.de/ moeller/racer/publications.html, July 2016.

[9] 2016. *FaCT++*; See https://code.google.com/archive/p/factplusplus/, July 2016.

[10] 2016. *Hermit*; See http://www.hermit-reasoner.com/, July 2016.

[11] 2016. *Geography Markup Language*; See http://www.opengeospatial.org/standards/gml, July 2016.

[12] A.A.C.U. *Ethical Reasoning Value Rubric*. Association of American Colleges & Universities(AACU), 2010.

[13] Abdelmoty A. I., Smart P. D., Jones C. B., Fu G., and Finch D. *A critical Evaluation of Ontology Languages for Geographic Information Retrieval on the Internet*. Journal of Visual Languages & Computing, Volume 16, Issue 4, Pages 331358, 2005.

[14] Abel D., Andersen T., and Conaboy C. *Jets crash on Logan taxiway*. In *Boston Globe, 10/03/16*.

[15] Afyouni I., Ray C., and Claramunt C. *Spatial models for context-aware indoor navigation systems: A survey. Journal of Spatial Information Science, Number 4 85123*, pages 85–123, 2012.

[16] Alexander H.G. *The Leibniz-Clarke Correspondence*. ed. by H.G.Alexander; Manchester,UK, 1956.

[17] Allen J. F., Frisch A. . *What's in a Semantic Network*. 20th. annual meeting of ACL, Toronto, pp. 19-27, 1982.

[18] Allen J.F. *Maintaining Knowledge about Temporal Intervals. Communications of the ACM*, 26(11):832–843, 1983.

[19] Allen J.F. *Towards a General Theory of Action and Time. Artificial Intelligence*, 23(2):123–154, 1984.

[20] Andrea C., Diego C., De Giacomo G., Maurizio L. *A Formal Framework for Reasoning on UML Class Diagrams. M.-S. Hacid et al. (Eds.): ISMIS 2002, LNAI 2366.*

[21] Annaswamy A. M., Soudbakhsh D., Schneider R., Goswami D., and Chakraborty S. *Arbitrated Network Control Systems: A Co-Design of Control and Platform for Cyber-Physical Systems.* Control of Cyber-Physical Systems: Lecture Notes in Control and Information Sciences, Volume 449, pp 339-356, 2013.

[22] Antsaklis P. *Defining Intelligent control: Report of the Task Force on Intelligent Control.* International Symposium on Intelligent Control, pp (i)-(xvii), Columbus, OH, August 16-18, 1994.

[23] Austin M.A., Baras J.S., and Kositsyna N.I. *Combined Research and Curriculum Development in Information-Centric Systems Engineering.* In *Proceedings of the Twelth Annual International Symposium of The International Council on Systems Engineering (INCOSE)*, Las Vegas, USA, July 2003.

[24] Austin M.A., Delgoshaei P., and Nguyen A. Distributed Systems Behavior Modeling with Ontologies, Rules, and Message Passing Mechanisms. In *Thirteenth Annual Conference on Systems Engineering Research (CSER 2015)*, Hoboken, New Jersey, March 17-19 2015.

[25] Austin M.A., Mayank V., and Shmunis N. PaladinRM: Graph-Based Visualization of Requirements Organized for Team-Based Design. *Systems Engineering: The Journal of the International Council on Systems Engineering*, 9(2):129–145, May 2006.

[26] Baader et al. *Description Logics.* In *Frank van Harmelen, Vladimir Lifschitz, and Bruce Porter, editors, Handbook of Knowledge Representation, chapter 3*, pages 135–180. Elsevier, 2008.

[27] Baader F. *Logic-based knowledge representation.* In Artificial intelligence today, PP. 13-41, Springer-Verlag Berlin, Heidelberg , 1999.

[28] Baader F., Horrocks I., and Sattler U. *Description Logics as Ontology Languages for the Semantic Web.* In *Dieter Hutter and Werner Stephan, editors, Mechanizing Mathematical Reasoning: Essays in Honor of Jrg Siekmann on the Occasion of His 60th Birthday, number 2605 in Lecture Notes in Artificial Intelligence*, pages 228–248. Springer, 2005.

[29] Baader F., McGuinness D.L. , Nardi D., and Patel-Schneider P.F. *The Description Logic Handbook: Theory, implementation, and applications.* Cambridge, 2003.

[30] Bateman J. A., Hois J., Ross R., and Tenbrink T. *A linguistic ontology of space for natural language processing.* Artificial Intelligence 174,10271071, 2010.

[31] Bateman J. and Farrar S. . *Spatial Ontology Baseline.* OntoSpace Project Report, SFB/TR 8: I1-[OntoSpace], Deliverable D2, Workpackage 1, University of Bremen, Germany, 2006.

[32] Baumgartner G., Heap D. and Krueger R. *Course notes for CSC165H: Mathematical expression and reasoning for computer science.* In *Department of Computer Science, University of Toronto, CA*, 2006.

[33] Beaubouef T., Petry F.E. and Ladner R. *Spatial data methods and vague regions: A rough set approach.* In *Applied Soft Computing, 7(1), 425440*, 2007.

[34] Berardi D., Diego C., and De Giacomo G. *Reasoning on UML class diagrams . Elsevier-Artificial Intelligence*, 168:70–118, 2005.

[35] Berners-Lee T., Hendler J., Lassa O. *The Semantic Web. Scientific American*, pages 35–43, May 2001.

[36] Bhave A., Krogh B., Garlan D., and Schmerl B. *Multi-domain Modeling of Cyber-Physical Systems Using Architectural Views.* Dept. of Electrical & Computer Engineering, Carnegie Mellon University, Pittsburgh, PA 15217, 2010.

[37] Bittner T. and Stell J. *Rough sets in approximate spatial reasoning.* In *W. Ziarko & Y. Yao, eds., Rough Sets and Current Trends in Computing, vol. 2005 of Lecture Notes in Computer Science, 445453, Springer Berlin / Heidelberg*, 2001.

[38] Bluman G.W. and Kumei S. *Symmetries and Differential Equations.* In *Springer-Verlag, New York*, 1989.

[39] Borgo S., Guarino N., and Masolo C. *A Pointless Theory of Space Based On Strong Connection and Congruence .* L. Carlucci Aiello and S. Shapiro (eds.): Proceedings of KR'96, Principles of Knowledge Representation and Reasoning, San Mateo, CA, USA, pp. 220-229, 1996.

[40] Brachman R. J. and Levesque H. J. . *The tractability of subsumption in frame-based description languages.* 4th National Conference of the American Association for Artificial Intelligence (AAAI-84). Austin, TX, pp. 34-37, 1984.

[41] Brooks C., Lee E.A., Liu X. , Neuendorffer S. , Zhao Y., and Zheng H. *Heterogeneous Concurrent Modeling and Design in Java (Volume 1: Introduction to Ptolemy II).* Technical Report ECB/EECS-2008-28, Department Electrical Engineering and Computer Sciences, University of California, Berkeley, CA, April 2008.

[42] Buckingham E. *On physically similar systems. PhysRev*, 4(4):345–376, 1914.

[43] Cardenas A.A., Saurabh A., and Shankar S. *Secure Control: Towards Survivable Cyber-Physical Systems.* LectureNotes/ss13/RN, University of California, Berkeley, California, USA, 2008.

[44] Carnegie Mellon University (CMU). Smart Traffic Signals. In *http://www.cmu.edu/homepage/computing/2012/fall/smart-traffic-signals.shtml; Accessed : November 15*, 2014.

[45] Cataldo A., Lee E. A., Liu X., Matsikoudis E., and Zheng H. *A constructive fixed-point theorem and the feedback semantic of timed systems.* Workshop on Discrete Event Systems (WODES), Ann Arbor, Michigan., Available from: http://ptolemy.eecs.berkeley.edu/publication/papers/06/constructive/, 2006.

[46] Chang M.S., Messer C.J., and Santiago A.J. *Timing traffic signal change intervals based on driver behavior.* Transportation Research Record, 1027, 20-30, 1985.

[47] Chein M., Mugnier M. L. *Graph-Based Knowledge Representation: Computational Foundations of Conceptual Graphs*. Springer Publishing Company, Incorporated, 1 edition, 2008, 2008.

[48] Choi N., Song I.Y. and Han H. *A survey on ontology mapping*. pages 34–41. SIGMOD Rec, 2006.

[49] Claramunt C. and Jiang B. *A Representation of Relationships in Temporal Spaces*. In: Innovations in GIS VII: Geocomputation, Taylor & Francis(eds.), pp. 41-53, 2000.

[50] Claramunt Christophe and Jiang Bin. *Hierarchical reasoning in space and time*. 9th International Symposium on Spatial Data Handling (SDH) Beijing, P.R.China, P. Forer, A. G. O. Yeh and J. He (eds.), pp. 3a. 41-51, 2000.

[51] Clarke B. *A Calculus of individual based on "connection"*. Notre Dame Journal of Formal Logic 22(3), 204-218, 1981.

[52] Clementini E. and Di Felice P. . *A spatial model for complex objects with a broad boundary supporting queries on uncertain data*. In *Data Knowledge and Engineering, 37(3), 285305*, 2001.

[53] Clementini E. and Di Felice P. *A Comparison of Methods for Representing Topological Relationships*. Information Sciences 3, 149-178, 1995.

[54] Cleveland State University. *Theories of accident causation*. Work Zone Safety and Efficiency Transportation Center, 2011.

[55] Cohen P. R. and Grinberg M. R. *A Theory of Heuristic Reasoning About Uncertainty*. AI Magazine Volume 4 Number 2 (1983) 17 - 24, 1983.

[56] Cohn A. G., Bennett B., Gooday J., and Gotts N. M. *Qualitative spatial representation and reasoning with the region connection calculus.* Geoinformatica,1:144, 1997.

[57] Console L., Dupre D. T., and Torasso P. *On the relationship between abdunction and deduction.* Journal of Logic Programming (1991) 1 , 661 690, 1991.

[58] Corcho O. and Gomez-Perez A. *A Roadmap to Ontology Specification Languages.* EKAW00. 12th International Conference on Knowledge Engineering and Knowledge Management, 2000.

[59] Corcho O., Fernandez-Lopez M., Gomez-Perez A., and Vicente O. *An Integrated Workbench for Ontology Representation, Reasoning and Exchange.* Prof. of EKAW2002, Springer LNAI 2473, 138-153, 2002.

[60] Davis M. *History of JTS and GEOS.* In *http://lin-ear-th-inking.blogspot.com/2007/06/history-of-jts-and-geos.html. Retrieved 2016-08-01* , 2007.

[61] Davis M. *Secrets of the JTS Topology Suite.* In *http://tsusiatsoftware.net/jts/files/. Retrieved 2016-08-01*, 2016.

[62] De Giacomo G. and Lenzerini M. *Concept Language With Number Restrictions and Fixpoints, and Its Relationship With Mu-Calculus.* 11th European Conference on Artificial Intelligence(ECAI), John Wiley and Sons, Ltd, 1994.

[63] Del Mondo G., Stell J. and Claramunt C. *A Graph Model for Spatio-temporal Evolution. Journal of Universal Computer Science*, 16:1452–1477, 2010.

[64] DeLaguna T. *Point, line and surface as sets of solids.* The journal of philosophy, 19, pp. 449-461, 1922.

[65] Delgoshaei P., Austin M.A., and Pertzborn A. *A Semantic Framework for Modeling and Simulation of Cyber-Physical Systems.* In *International Journal On Advances in Systems and Measurements, Vol. 7, No. 3-4, December, 2014, pp. 223–238.*, 2014.

[66] Derler P., Lee Edward A. and Sangiovanni-Vincentelli A.L. *Addressing Modeling challenges in Cyber-Physical Systems.* Technical Report N0 . UCB/EECS-2011-17, Electrical Engineering and Computer Sciences University of California Berkley, 2011.

[67] Dilo A., De By R. and Stein. *A system of types and operators for handling vague spatial objects.* In *International Journal of Geographical Information Science, 21(4), 397426*, 2007.

[68] Dowden B. *Time.* In *Internet Encyclopedia of Philosophy, available at http://www.iep.utm.edu/time/, accessed 07/11/2016*, 2016.

[69] Doyle J. *Feedback Control Theory.* class notes, CDS 212 Fall 2011, Available at: https://www.cds.caltech.edu/wiki/index.php/, 2011.

[70] Elder L. and Paul R. *Thinker's Guide to Understanding the Foundations of Ethical Reasoning.* Foundation for Critical Thinking, 2013.

[71] Energetics Incorporated. *Cyber-physical Systems Situation Analysis of Current Trends, Technologies, and Challenges.* Energetics Incorporated for the National Institute of Standards and Technology (NIST), 2012.

[72] Ermolayev V., Batsakis S, Keberle N., Tatarintseva O., and Antoniou G. *Ontologies of time: review and trends* . International Journal of Computer Science and Applications,Technomathematics Research Foundation, Vol. 11, No. 3, pp. 57 115, 2014.

[73] Ermolayev V., Keberle N., Tatarintseva O., Matzke W-E, and Sohniusand R. *Fuzzy Time Intervals for Simulating Actions* . Information Systems and e-Business Technologies, Volume 5 of the series Lecture Notes in Business Information Processing, pp. 429-444, 2008.

[74] European Union. *The ARTEMIS Embedded Computing Systems Initiative.* ARTEMIS joint Undertaking, Available at: http://www.artemis-ju.eu/, 2012.

[75] Feron E. and Balakrishnan H. *CPS and NextGen: Cyber-Physical Systems Challenges in Next Generation Aviation.* 2010 NITRD High Confidence Software Systems (HCSS) Group, 2011.

[76] Fikes R. and Kehler T. . *The Role of Frame-based Representation in Reasoning.* In *Communications of the ACM, Special edition, Volume 28 Number 9, PP. 904-920* , 1985.

[77] Fikes R. and Zhou Q. *A Reusable Time Ontology.* In *AAAI Technical Report WS-02-11, AAAI (www.aaai.org)*, 2002.

[78] Fisher M. In *An Introduction to Practical Formal Methods Using Temporal Logic.* John Wiley & Sons, Ltd, 2011.

[79] Fodor G. A. *Ontology Controlled Autonomous System Principles, Operations and Architecture.* Kluwer Academic Publishers, 1998.

[80] Foudil C., Noureddine N., Sanza C., and Duthen Y. *Path Finding and Collision Avoidance in Crowd Simulation.* In *Journal of Computing and Information Technology - CIT 17, 3, pp.217228*, 2009.

[81] Francisco J., Silva M. J., and Chaves M. *Linkable Geographic Ontologies.* 6th Workshop on Geographic Information Retrieval(GIR 10), Zurich, Switzerland, 18-19th Feb., 2010.

[82] Freksa C. *Communication about visual patterns by means of fuzzy characterizations.* In *In $XXII^{nd}$ International Congress of Psychology, Leipzig, Germany*, 1980.

[83] Freksa C. *Fuzzy systems in AI: An overview.* In *In R. Kruse, J. Gebhardt & R. Palm, eds., Fuzzy systems in computer science, 155169, Vieweg, Braunschweig/Wiesbaden*, 1994.

[84] Freksa C. and De Mantaras L. *An adaptive computer system for linguistic categorization of soft observations in expert systems and in the social sciences.* In *2nd World Conference on Mathematics at the Service of Man, PP.288292, Las Palmas, Spain*, 1982.

[85] Fridenthal S., Moore A., and Steiner R. *A Practical Guide to SysML.* MK-OMG, 2008.

[86] Frigg R. and Hartmann S. *Scientific Models. The Philosophy of Science-An Encyclopedia, N-Z index, Sahotra Sarkar and Jessica Pfeifer(eds)*, 2:740–749, 2006.

[87] Furia C.A., Mandrioli D., Morzenti A., and Rossi M. Modeling Time in Computing: A Taxonomy and a Comparative Survey. *ACM Computing Surveys*, 42(2), February 2010.

[88] Galindo C., Saffiotti A., Coradeschi S., Buschka P., Fernandez-Madrigal J.A. and Gonzalez J. *Multi-Hierarchical Semantic Maps for Mobile Robotics.* EEE/RSJ International Conference on Intelligent Robots and Systems (IROS-05), Edmonton, Canada, August, 2005.

[89] Galton A. *Taking dimension seriously in qualitative reasoning.* W. Wahlster (ed), ECAI'96. Chichester, pp. 501-505, 1996.

[90] Gamma E., Helm R., Johnson R., and Vlissides J. *Design Patterns: Elements of Reusable Object-Oriented Software.* Addison-Wesley Professional Computing Series, 1995.

[91] Gasos J. and Saffiotti A. *Using fuzzy sets to represent uncertain spatial knowledge in autonomous robots.* In *IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems, pp.420 - 425, Heidelberg, Germany*, 1999.

[92] General Motors Corporation (GMC). *GM Unveils EN-V Concept: A Vision for Future Urban Mobility.* In *http://media.gm.com; Accessed : November 15*, 2015.

[93] George B. and Shekhar S. *Time-Aggregated Graphs for Modeling Spatio-temporal Networks.* Advances in Conceptual Modeling - Theory and Practice Lecture Notes in Computer Science Volume 4231, Springer(eds) pp 85-99, 2006.

[94] Geroimenko V., and Chen C. (Eds). *Visualizing the Semantic Web: XML-based Internet and Information Visualization.* Springer, 2003.

[95] Giannakopoulou D. *NASAs State-Space Exploration: Verifying Safety-Critical Systems.* In *CMU / NASA Ames Research Center*, 2009.

[96] Glasgow J. *The Imagery Debate Revisited: A Computational Perspective. Computational Intelligence*, 9:309–333, 1993.

[97] Golbreich C., Zhang S., and Bodenreider O. *The foundational model of anatomy in OWL: Experience and perspectives.* Journal of Web Semantics, 4(3), 2006.

[98] Goodchild Michael F. *GIS as a Sandbox: The Challenge of Spatio-Temporal Analysis and Modeling.* Space-Time Modeling and analysis Workshop, February 22-23, 2010, Redland, CA, USA , 2010.

[99] Goodwin J. *Experiences of using OWL at the ordnance surveys.* First OWL Experiences and Directions Workshop, volume 188 of CEUR Workshop Proceedings. CEUR(http://ceur-ws.org/), 2005.

[100] Google inc. *The Latest Chapter for the Self-Driving Car: Mastering City Street Driving.* In *Official Google Blog. N.p., n.d. Web. April, 28 2014, http://googleblog.blogspot.com/2014/04/the-latest-chapter-for-self-driving-car.html; Accessed : November 15*, 2015.

[101] Gotts N. M. *Formalizing Commonsense Topology: The INCH Calculus.* Four International Symposium on Artificial Intelligence and Mathematics - AI/MATH'96, Fort Lauderdale(FL), pp. 72-75, 1996.

[102] Grau et al. *OWL 2: The next step for OWL. Journal of Web Semantics: Science, Services and Agents on the World Wide Web*, 6(4):309322, 2008.

[103] Graves H. *Integrating SysML and OWL .* OWL: Experiences and Directions (OWLED), 2009.

[104] Graves H. *Current state of ontology in engineering systems*. OMG: Ontology Action Team, Available from: www.omgwiki.org/MBSE/oku.php?id, 2012.

[105] Graves H. *Integrating Reasoning with SysML*. INCOSE International Symposium, Rome, Italy, 2012.

[106] Graves H. and Bijan Y. *Using formal methods with SysML in aerospace design and engineering* . Annals of Mathematics and Artificial Intelligence, Springer , 2011.

[107] Groger G., Plumer L. *CityGML Interoperable semantic 3D city models*. ISPRS Journal of Photogrammetry and Remote Sensing, Volume 71, Pages 1233, 2012.

[108] Gruber T. R. and Olsen G. R. *An Ontology for Engineering Mathematics*. In *In Doyle, Torasso, and Sandewall, Eds., Fourth International Conference on Principles of Knowledge Representation and Reasoning*. Morgan Kaufman, 1994.

[109] Gruninger M. *Ontology of the Process Specification Language*. In Handbook on Ontologies, Staab et al.(eds), 2004.

[110] Gusgen H. W. *Spatial reasoning based on Allen's temporal logic*. Report ICSI TR-89-049, International Computer Science Institute, Berkeley, CA, USA, 1989.

[111] Habel C. *Propositional and Depictorial Representations of Spatial Knowledge: The case of Path Concepts*. R. Studer(ed.): Natural Language and Logic, Lecture Notes in Computer Science,Berlin:Springer Verlag, pp 94-117, 1990.

[112] Haghighi M. M. *Controlling Energy-Efficient Buildings in the Context of Smart Grid: A Cyber- Physical Systems Approach*. Technical Report UCB/EECS-2013-244, Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720, USA, 2013.

[113] Hanche-Olsen H. *Buckinghams pi-theorem.* In *TMA4195 Mathematical modelling*, 2004.

[114] Hatala M., Wakkary R. and Kalantari L. *Rules and Ontologies in support of Real-Time Ubiquitous Application. Journal of Web Semantics*, 3:5–22, 2005.

[115] Hayes P. *A Catalog of Temporal Theories.* Technical Report UIUC-BI-AI-96-01, University of Illinois, USA, 1994.

[116] Hayes P. J. *The logic of frames.* D. Metzing, Ed., Frame Conceptions and Text Understanding. Berlin: deGruyter, pp. 46-61, 1980.

[117] Hendler J. *Agents and the Semantic Web. IEEE Intelligent Systems*, pages 30–37, March/April 2001. Available on April 4, 2002 from http://www.computer.org/intelligent.

[118] Hernandez D., Clementini E. and Di Felice P. *Qualitative Distances* . A. Frank and W. Huhn (eds.): Spatial Information Theory, Proceedings of COSIT'95. Berlin, pp. 45-57, 1995.

[119] Hertkorn P. and Rudolph S. *Dimensional Analysis in Case-Based Reasoning.* In *Proceedings International Workshop on Similar Methods*, 1998.

[120] Hobbs J. R. *DAML Space - An ontology of spatial relations for the semantic web.* Invited talk, Workshop on the Analysis of Geographic References, Edmonton, Canada, 2003.

[121] Hobbs J. R. and Pan F. *An ontology of time for the Semantic Web. Proceeding-ACM transactions on Asian Language processing(TALIP)*, 3(1):66–85, 2004.

[122] Hoffmann G. M. and Tomlin C. J. *Decentralized Cooperative Collision Avoidance for Acceleration Constrained Vehicles*. In *Proceedings of the 47th IEEE Conference on Decision and Control Cancun, Mexico, Dec. 9-11*, 2008.

[123] Hooman J. *Introduction to Timed Systems*. University of Nijmegen, Netherlands, November 2001.

[124] Horridge M., Jupp S., Moulton G. Rector A., Stevens R., and Wroe C. A Practical Guide to Building OWL Ontologies using Protege 4 and CO-ODE Tools, October 2007. University of Manchester, England.

[125] Horrocks I. *Description Logic Reasoning*. The University of Manchester, 2005.

[126] Horrocks I. *Ontologies and the Semantic Web*. Communications of the ACM, 51(12):58-67, December, 2008.

[127] Horrocks I. et al. *A description logic with transitive and inverse roles and role hierarchies. Journal of Logic and Computation*, 9(3):385–410, 1999.

[128] Horrocks I., Kutz O., and Sattler U. *The Even More irresistible SROIQ*. pages 57–67. 10th International Conference on Principles of Knowledge Representation and Reasoning (KR 2006), AAAI Press, 2006.

[129] Horrocks I., Patel-Schneider P. F., and Van Harmelen F. *From SHIQ and RDF to OWL: The Making of a Web Ontology Language*. Journal of Web Semantics, 1(1):7-26, 2003.

[130] Hurwitz D. S. *The "Twilight Zone" of Traffic costs lives at Stoplight Intersections*. Oregon State University, Corvallis, Oregon, USA, 2012.

[131] Hurwitz D. S., Wang H. B., Knodler M. A., Nib D., and Moorea D. *Fuzzy Sets to Describe Driver Behavior in the Dilemma Zone of High-Speed Signalized Intersections.* School of Civil and Construction Engineering, Oregon State University, USA and Department of Civil and Environmental Engineering, University of Massachusetts Amherst, USA, 2012.

[132] Huynh T., Alsadah A. and Hu F. *Cyber-Physical System Controls.* Cyber-Physical Systems: Integrated Computing and Engineering Design, 2010.

[133] International Business Machine (IBM) Corporation. *IBM Intelligent Transportation Solution for Active Traffic Management.* In *Systems and Technology Group, IBM Corporation, November*, 2013.

[134] Jackson D. *Dependable Software by Design. Scientific American*, 294(6), June 2006.

[135] Jensen J.C., Chang D.H. and Lee E.A. *A Model-Based Design Methodology for Cyber-Physical Systems.* IEEE Workshop on Design, modeling, and Evaluation of Cyber-Physical Systems(CyPhy), Istanbul, Turkey, 2011.

[136] JPL Special Review Board. *Report on the loss of the Mars polar lander and deep space 2 missions.* NASA Jet Propulsion Laboratory, 2000.

[137] Junger E. *Symbolic Spatial Reasoning on Object Shapes for Qualitative Matching.* A. Frank and I. Campari(eds.) Spatial Information Theory: A theoretical basis for GIS, COSIT'93, LNCS No. 716, Springer-Verlag, pp. 444-462, 1993.

[138] Kitching C. *2 planes clip on Metro Airport taxiway.* In *Daily Mail, Available at:* $http://www.dailymail.co.uk/travel/travel_news/article-3021054/Part-wing-$

torn−two−Ryanair−planes−collide−taxiing−runway−Dublin−Airport.html, accessed 10/03/16, 2016.

[139] Klai S., and Kahdir M.T. Approach for a Rule Base Ontologies Integration. In *5th International Conference on Computer Science and Information Technology*, 2013.

[140] Kopetz H. Event-triggered versus Time-triggered Real-Time Systems. *International Workshop on Operating Systems of the 90s and beyond, Lecture Notes in Computer Science*, 563():87–101.

[141] Krebs B. *Cyber Incident Blamed for Nuclear Power Plant Shutdown* . Washington Post, June 5, 2008.

[142] Krieger H. *A General Methodology for equipping Ontologies With Time*. German Research Center for Artificial Intelligence(DFKI), 2009.

[143] Krotzsch M., Simanck F. and Horrocks I. *A Description Logic Primer*. Department of computer Science, University of Oxford, UK, Version 1.2, 2013.

[144] Kuipers B. *An Ontological Hierarchy for Spatial Knowledge*. AAAI Technical Report FS-94-03, 1994.

[145] Kuipers B. *The Spatial Semantic Hierarchy*. Artificial Intelligence, 119:191233, 2000.

[146] Kuipers B. *An Intellectual History of the Spatial Semantic Hierarchy*. Springer Tracts in Advanced Robotics, 38:243264, 2008, 2008.

[147] Lacher A. R., Maroney D. R., and Zeitlin A. D. *Unmanned aircraft collision avoidance technology assessment and evaluation methods*. In *The MITRE Corporation, McLean, VA, USA*, 2007.

[148] Lacy L., Aviles G., Fraser K., Gerber W., Mulvehill A., and Gaskill R. *Experiences of using OWL in military applications.* First OWL Experiences and Directions Workshop, volume 188 of CEUR Workshop Proceedings. CEUR(http://ceur-ws.org/), 2005.

[149] Lampson B. *Getting Computers to Understand.* Journal of the Association for Computing Machinery(JACM), v50, n1, pp.70-72, 2003.

[150] Latecki L. and Pribbenow S. *On Hybrid Reasoning for Processing Spatial Expressions.* ECAI' 92 pp. 389-393, 1992.

[151] Laurini R. *Urban ontologies.* Workshop on urban ontologies, INSA, Lyon, 2004.

[152] Layman L., Basili V. R., and Zelkowitz M. V. *The Role and Quality of Software Safety in the NASA Constellation Program.* Technical Report 10-101, 2010.

[153] Lee E. A. *The Problem with Threads.* EECS Department, University of California at Berkeley, Berkeley, CA, 2006.

[154] Lee E. A. *Cyber-Physical Systems : Design Challenges.* Technical Report No. UCB/EECS-2008-8 , 2008.

[155] Lee E. A. *Time is a Resource, and Other Stories.* International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing(ISORC), Orlando, FL, USA, 2008.

[156] Lee E. A. *Computing needs time. Communications of the ACM*, 52:70–79, 2009.

[157] Lee E. A. *CPS Foundations.* DAC10, Anaheim, California, USA, 2010.

[158] Lee E. A. *Cyber-Physical Systems : A Rehash or A New Intellectual Challenge?* Invited Talk in the Distinguished Speaker Series, Design Automation Conference (DAC), Austin, TX, 2013.

[159] Lehmann F. and Cohn A.G. *The egg/yolk reliability hierarchy: semantic data integration using sorts with prototypes.* In *Third International Conference on Information and Knowledge Management, CIKM 94, 272279, ACM*, 1994.

[160] Leveson N. G. *A Systems-Theoretic Approach to Safety in Software-Intensive Systems. IEEE Transactions on Dependable and Secure Computing*, 1:66–86, 2004.

[161] Leveson N. G. *The Role of Software in Spacecraft Accidents. AIAA Journal of Spacecraft and Rockets*, 41:564–575, 2004.

[162] Leveson N. G. *Engineering a Safer World: Systems Thinking applied to Safety.* MIT Press, 2011.

[163] Leveson N.G. *A New Approach to Software Systems Safety Engineering. System Safety Engineering: Back to the Future. See http://sunnyday.mit.edu/book2.html*, 2006.

[164] Levin D. P. *Collision in Detroit; At Least 8 Die in Collision On Detroit Airport Runway.* In *The New York Times, Available at: http://www.nytimes.com/1990/12/04/us/collision-in-detroit-at-least-8-die-in-collision-on-detroit-airport-runway.html, accessed 10/03/16*, 1990.

[165] Leyton M. *A process grammar for shape.* Artificial Intelligence, 34, pp. 59-68, 1988.

[166] Li P. *Stochastic Methods for Dilemma Zone Protection at Signalized Intersections.* Doctor of Philosophy Dissertation submitted to the faculty of the Virginia Polytechnic Institute and State University, VA, USA , 2009.

[167] Li Z., Wang Z., Zhang A., and Xu Y. The Domain Ontology and Domain Rules Based Requirements Model Checking. *International Journal of Software Engineering and Its applications*, 1(1), July 2007.

[168] Linkov J. *Collision-Avoidance Systems Are Changing the Look of Car Safety.* In *Consumer Reports, December 2015, Avaliable at : http://www.consumerreports.org/car-safety/collision-avoidance-systems-are-changing-the-look-of-car-safety/*, 2015.

[169] Liu K. and Shi W. *Computing the fuzzy topological relations of spatial objects based on induced fuzzy topology.* In *International Journal of Geographical Information Science, 20(8), 857883*, 2006.

[170] Loos S., Platzer A. and Nistor L. *Adaptive Cruise Control: Hybrid, Distributed, and Now Formally Verified.* In *CMU-CS-11-107, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA, June*, 2015.

[171] Lowy J. and Krisher T. *Tesla driver killed in crash while using car's "Autopilot".* In *Associated Press, available at https://www.yahoo.com/news/self-driving-car-driver-died-205642937.html, accessed 07/01/2016*, 2016.

[172] Lutz C. *The Complexity of Reasoning with Concrete Domains.* LTCS-Report 99-01, Aachen university of Technology Research group for Theoretical Computer Science, 1999.

[173] Ma J., Knight B., Petridis M., and Bai X. *A Graphical Representation for Uncertain and Incomplete Temporal Knowledge*. In *Second WRI Global Congress on Intelligent Systems, Wuhan, China*, 2010.

[174] Maas D. *Dilemma zone elimination*. Sacramento Department of Transportation, Sacramento, CA, USA, 2008.

[175] Machol R. E. *The Titanic coincidence*. *Interfaces*, 5:53–54, 1975.

[176] Mahmoud Q.H. Getting Started With the Java Rule Engine API (JSR 94): Toward Rule-Based Applications, 2005. Sun Microsystems. For more information, see http://java.sun.com/developer/technicalArticles/J2SE/JavaRule.html (Accessed, March 10, 2008).

[177] Manna Z. and Pnueli A. *The Temporal Logic of Reactive and Concurrent Systems*. Springer, Berlin, 1992.

[178] Marwedel P. *Embedded System Design: Embedded systems Foundations of Cyber-Physical Systems*. 2nd Edition, Springer, 2011.

[179] Masolo C., Borgo S., Gangemi A., Guarino N., Oltramari A., and Schneider L. *WonderWeb Deliverable D17 - The WonderWeb Library of Foundational Ontologies - Preliminary Report*. IST Project 2001-33052 WonderWeb: Ontology Infrastructure for the Semantic Web, Commission of the European Communities, 2003.

[180] Menzies T. *Applications of Abduction: Knowledge-Level Modeling*. Int . J . Human Computer Studies (1996) 45 , 305 335, 1996.

[181] Minkel J. *The 2003 Northeast Blackout – Five Years Later*. Scientific American, 2008.

[182] Minkowski H. *Raum und Zeit*. Physikalische Zeitschrift 10: 7588, 1908.

[183] Minsky M. *A Framework for Representing Knowledge*. Technical Report, MIT-AI Laboratory, Massachusetts Institute of Technology Cambridge, MA, USA, 1974.

[184] Mosteller M., Austin M.A., Yang S., and Ghodssi R. *Platforms for Engineering Experimental Biomedical Systems. IEEE Systems Journal*, PP(9):1–11, September 2014.

[185] Moszkowski B. and Manna Z. *Reasoning in interval Temporal Logic. Proc. AMC/NCF/ONR Workshop on Logics of Programs*, 164:371–383, 1984.

[186] Muller D. Requirements Engineering Knowledge Management based on STEP AP233. 2003.

[187] Muller P. *Topological spatio-temporal reasoning and representation*. IRIT-Universite Paul Sabatier, 31062 Toulouse, France, 2002.

[188] Muskerjee A. and Joe G. *A qualitative Model for Space*. Association for the Advancement of Artificial Intelligence(AAAI), Conference on Artificial Intelligence (AAAI-90), Cambridge, MA, USA, pp. 721-727, 1990.

[189] Myers C. *Modeling and Verification of Cyber-Physical Systems*. Design Automation Summer School, University of Utah, 2011.

[190] Nagypal G. and Motik B. *A Fuzzy Model for Representing Uncertain, Subjective and Vague Temporal Knowledge in Ontologies*. In *FZI Research Center for Information Technologies, University of Karlsruhe, Germany*, 2003.

[191] National Institute of Standards and Technology (NIST). *Strategic R & D Opportunities for 21st Cyber-physical Systems: Connecting computer and information sys-*

tems with the physical world. National Institute of Science and Technology(NIST), Gaithersburg, MD, USA , 2013.

[192] National Science Foundation(NSF). *Report: Cyber-Physical Systems Summit* . CPS Week, April 24-25, 2008, St Louis, Missouri, MS, USA, 2008.

[193] Nigel C. *Computability: An Introduction to Recursive function Theory*. Cambridge, MA:Cambridge University Press, 1997.

[194] NITRD. *High-Confidence Medical Devices: Cyber-Physical Systems for 21st Century Health Care*. The Networking and Information Technology Research and Development (NITRD) Program, 2009.

[195] NITRD. *Winning the Future with Science and Technology for 21st Century Smart Systems*. Networking and Information Technology Research and Development(NITRD), Arlington, VA, USA, 2011.

[196] Olivetti N. *Artificial Intelligence: Introduction to Description Logics*. INCA-LSIS, Paul Cezanne university, Marseille, France, 2009.

[197] Open Street Map(OSM). https://www.openstreetmap.org, Accessed May,14. 2015.

[198] Pant P. D. and Cheng Y. . *Dilemma zone protection and signal coordination at closely-spaced high-speed intersections*. Report FHWA/OH-2001/12, Ohio Department of Transportation, Columbus, OH, USA, 2001.

[199] Pappas G. *Cyber-Physical Systems Research Challenges*. Penn Research in Embedded Computing and Integrated Systems Engineering (PRECISE), 2010.

[200] Pate-Cornell E. *Organizational aspects of engineering system safety: The case of offshore platforms. Science*, 250:12101217, 1990.

[201] Patel-Schneider P. F., Hayes P., and Horrocks I. *OWL Web Ontology Language semantics and abstract syntax.* Recommendation, W3C, Available at http://www.w3.org/TR/owl-semantics/, 2004.

[202] Pauly A. and Schneider M. *Spatial vagueness and imprecision in databases.* In *In R.L. Wainwright & H. Haddad, eds., Proceedings of the 2008 ACM Symposium on Applied Computing (SAC), 875879, ACM*, 2008.

[203] Pavlic M., Mestrovic A., Jakupovic A. *Graph-Based Formalisms for Knowledge Representation.* 17th World Multi-Conference on Systemics, Cybernetics and Informatics, July 9 -12, Orlando, Florida, USA, 2013.

[204] PCAST. *Ensuring American Leadership in Advanced Manufacturing.* Executive Office of the President, President's Council of Advisors on Science and Technology(PCAST) , 2011.

[205] PCAST. *Capturing Domestic Competitive Advantage in Advanced Manufacturing.* Executive Office of the President, President's Council of Advisors on Science and Technology(PCAST) , 2012.

[206] Petnga L. and Austin M.A. *Cyber-Physical Architecture for Modeling and Enhanced Operations of Connected-Vehicle Systems.* 2nd International Conference on Connected Vehicles & Expo (ICCVE 2013), Las Vegas, NV, USA, 2013.

[207] Petnga L. and Austin M.A. *Ontologies of Time and Time-based Reasoning for MBSE of Cyber-Physical Systems.* 11th Annual Conference on Systems Engineering Research (CSER 2013), Georgia Institute of Technology, Atlanta, GA, 2013.

[208] Petnga L. and Austin M.A. *Model-Based Systems Engineering for Design and Automated Operation of Modern Waterway Systems.* In *8th Annual IEEE International Systems Conference (SysCon 2014). Ottawa, Canada*, March 31 – April 3 2014.

[209] Petnga L. and Austin M.A. *Semantic Platforms for Cyber-Physical Systems.* 24th Annual International Symposium of The International Council on Systems Engineering (INCOSE), Las Vegas, NV, USA, June 30 - July 03, 2014.

[210] Petnga L. and Austin M.A. *Spatial Modeling and Reasoning for Cyber-Physical Systems.* In *International Conference on Complex Systems Engineering (ICCSE), Storrs, CT, USA, November 09-11*, 2015.

[211] Petnga L. and Austin M.A. *Tubes and Metrics for Solving the Dilemma-Zone Problem.* In *The Tenth International Conference on Systems(ICONS 2015), Barcelona, Spain, April 19 - 24*, pages 119–124, 2015.

[212] Petnga L. and Austin M.A. *An Ontological Framework for Knowledge Modeling and Decision Support in Cyber-Physical Systems. Advanced Engineering Informatics*, 30(1):77–94, January 2016.

[213] Petnga L. and Austin M.A. *Model-Based Design and Formal Verification Processes for Automated Waterway System Operations. Systems: Special Issue on Product, Process, System Design Review Methods and Tools*, 4(2):1–23, June 2016.

[214] Petnga L. and Xu H. *Security of Unmanned Aerial Vehicles: Dynamic State Estimation under Cyber-Physical Attacks.* In 4^{th} *International Conference on Unmanned Aircraft Systems (ICUAS 2016), Arlington, VA*, June 7-10 2016.

[215] Petrov V. *Process ontology in the context of applied philosophy.* In Vesselin Petrov, ed. Ontological Landscapes: Recent Thought on Conceptual Interfaces Between Science and Philosophy. Ontos Verlag. pp. 137 ff. ISBN 3868381074 , 2011.

[216] Platzer A. *Quantified differential dynamic logic for distributed hybrid systems.* In *In Anuj Dawar and Helmut Veith, editors, CSL, volume 6247 of LNCS, pages 469 483. Springer*, 2010.

[217] Podobnikar T. and Ceh M. *A linguistic ontology of space for natural language processing.* IGI Global, 304 pages, doi:10.4018/978-1-4666-0327-1, 2012.

[218] Poovendran R. et al. *A Community Report of the 2008 High Confidence Transportation Cyber-Physical Systems.* High Confidence Transportation Cyber-Physical Systems(HCTCPS) Workshop, Washington, DC, USA, 2009.

[219] Portinale L. *Modeling Uncertain Temporal Evolutions in Model-Based Diagnosis* . In *Eighth Conference on Uncertainty in Artificial Intelligence (UAI1992), Stanford, California, USA*, 1992.

[220] Praitosh P. K. *The Heuristic Reasoning Manifesto.* Qualitative Reasoning Group, Electrical and Computer Science, Northwestern University, Evanston IL 60208, USA, 2006.

[221] Ptolemaeus C., Editor. *System Design, Modeling, and Simulation using Ptolemy II.* In *Ptolemy.org*, 2014.

[222] Quillian M. *Semantic memory.* M. Minsky (Ed.), Semantic Information Processing, pp. 227-270, MIT, 1968.

[223] Rasmussen J. *Human error and the problem of causality in analysis of accidents.* ed. D. E. Broadbent, J. Reason, and A. Baddeley, 112. Oxford: Clarendon Press, 1990.

[224] Reason J. *Managing the Risks of Organizational Accidents.* London: Ashgate, 1997.

[225] Renz J. and Nebel B. *On the Complexity of Qualitative Spatial Reasoning: A Maximal Tractable Fragment of the Region Connection Calculus. Artificial Intelligence*, 108:69–123, 1997.

[226] Richards W. and Hoffman D. D. *Codon constraints on closed 2D shapes.* Computer vision, Graphics and Image Processing, 31, pp. 265-281, 1985.

[227] Rohrig R. *A theory of Qualitative Spatial Reasoning based on order relations.* 2th American Confernce on AI(AAAI-94), pp. 1418-1423, 1994.

[228] Rouquette N. and Jenkins S. *Transforming OWL2 Ontologies into Profiles Extending SysML* . 12th NASA-EST Workshop on Product Data Exchange, Oslo, Norway, 2010.

[229] Rudolf G. Some Guidelines For Deciding Whether To Use A Rules Engine, 2003. Sandia National Labs. For more information see http://herzberg.ca.sandia.gov/guidelines.shtml (Accessed, March 10, 2008).

[230] Rudolph S. *Knowledge discovery in scientific data.* In *Proceedings SPIE Conference DMKD, Florida, USA*, 2000.

[231] Russell S. and Norvig P. *Artificial Intelligence, A Modern Approach.* Prentice Hall, second edition, 2003.

[232] Sabet P. G. P., Aadal H., Mir H. M. J., Kiyanoosh G. R. *Application of Domino Theory to Justify and Prevent Accident Occurance in Construction Sites. IOSR Journal of Mechanical and Civil Engineering (IOSR-JMCE)* , 6:72–76, 2013.

[233] Sanjiang L. and Mingsheng Y. *Region Connection Calculus: its models and composition table. Artificial Intelligence*, 145:121–146, 2003.

[234] Schild K. *Terminological Cycles and the Propositional With Mu-Calculus.* In *4th Int. Conference on the Principle of Knowledge Representation and Reasoning(KR-94)*, pages 509–520. J. Doyle, E. Sandewall, and P. Torasso, 1994.

[235] Schlieder C. *Reasoning about ordering* . A. Frank and I. Campari (eds.): Spatial Information Theory - A Theoretical Basis for GIS, Proceedings of COSIT'95. Berlin, pp. 341-349, 1995.

[236] Schulz S. and Hahn U. *Mereotopological Reasoning about Parts and (W)Holes in Bio-Ontologiest.* FOIS 01, Ogunquit, Maine, USA, October 17-19, 2001.

[237] Segaran T., Taylor J., Evans C. *Programming the Semantic Web.* O'Reilly, Beijing, 2009.

[238] Sidhu A., Dillon T., Chang E., and Sidhu B. S. *Protein ontology development using OWL.* First OWL Experiences and Directions Workshop, volume 188 of CEUR Workshop Proceedings. CEUR(http://ceur-ws.org/), 2005.

[239] Sidorova N. Lecture Notes in Process Modeling. 2007. Department of Mathematics and Computer Science, Eindhoven University, Netherlands.

[240] Siegemund K., Thomas E.J., Zhao Y, Pan J., and Assmann U. Towards Ontology-Driven Requirements Engineering. In *7th International Workshop on Semantic Web Enabled Software Engineering*, 2011.

[241] Siemens Corporation. *Intelligent traffic solutions.* In *http://www.siemens.com; Accessed : November 15*, 2015.

[242] Sipser M. *The Halting Problem.* In *Introduction to the Theory of computation(Second Edition ed.). Section 4.2.* . PWS Publishing pp. 173-182. ISBN 0-534-94728-X, 2005.

[243] Sirin et al. *Pellet: A Practical OWL-DL Reasoner.* MIND Lab, University of Maryland, College Park MD 20742, USA, 2007.

[244] Skrzypczynski P. *Merging Probabilistic and Fuzzy Frameworks for Uncertain Spatial Knowledge Modelling.* In *4th International Conference on Computer Recognition Systems, CORES'05, Rydzyna Castle, Poland*, 2005.

[245] Skrzypczynski P. *Uncertain Spatial Knowledge Management in a Mobile Robot Architecture.* In *Spatial Cognition and Computation, Volume 1, Issue 3, pp 205-226*, 2006.

[246] Smart Manufacturing Leadership Coalition(SMLC). *Implementing 21st century Smart Manufacturing.* Workshop Summary Report, 2011.

[247] Smith B. *Ontology and the Logistic Analysis of Reality.* N. Guarino and R. Poli (eds): International Workshop on Formal Ontology in Conceptual Analysis and Representation pp. 51-68, 1993.

[248] Smith B. and Grenon P. *The cornucopia of formal-ontological relations. Dialectica*, 58:279 296, 2004.

[249] Smith M. *NASA's Space Shuttle Columbia: Synopsis of the Report of the Columbia Accident Investigation Board*. CRS Report for Congress, 2003.

[250] Smith R., Self M. and Cheeseman P. *Estimating Uncertain Spatial Relationships in Robotics*. In *Second Conference on Uncertainty in Artificial Intelligence (UAI1986)* , 1986.

[251] Soergel D., Lauser B., Liang A., Fisseha F., Keizer J., and Katz S. *Reengineering thesauri for new applications: The AGROVOC example*. Journal of Digital Information, 4(4), 2004.

[252] Song F., Zacharewicz G. and Chen D. *An ontology-driven framework towards building enterprise semantic information layer*. Advanced Engineering Informatics (27) 3850, Elsevier, 2013.

[253] Sonin A. A. *The Physical Basis of Dimensional Analysis*. 2nd Edition, Ain A. Sonin, 2001.

[254] Sowa J. F. *Conceptual Structures: Information Processing in Mind and Machine*. Addison-Wesley. ISBN 978-0-201-14472-7, 1984.

[255] Sowa J. F. *Conceptual Graphs*. In *In Handbook of Knowledge Representation, Chapter 5, ed. by F. van Harmelen, V. Lifschitz, and B. Porter, Elsevier, pp. 213237* , 2008.

[256] Sowa J. F., Borgida A. . *Principles of Semantic Networks: Explorations in the Representation of Knowledge*. John F. Sowa (eds), 1991.

[257] Starren J. and Xie G. *Comparison of Three Knowledge Representation Formalisms, for Encoding the NCEP Cholesterol Guidelines*. Center for Medical Informatics, Columbia University College of Physicians and Surgeons, New York, 1994.

[258] Strobl C. *Dimensionally Extended Nine-Intersection Model (DE-9IM)*. In *Encyclopedia of GIS, Springer US, pp 240-245* , 2008.

[259] Sunanda B. and Seetharamaiah P. *Modeling of Safety-Critical Systems Using Petri Nets*. In *ACM SIGSOFT Software Engineering Notes archive, Volume 40 Issue 1, January 2015, pp. 1–7*, 2015.

[260] Swarup M. B. and Ramaiah P. S. A Software Safety Model for Safety Critical Applications. *International Journal of Software Engineering and its Applications*, 3:21–32, 2009.

[261] Sztipanovits J. *Model Integration Challenges in Cyber-Physical Systems. A short course to NIST Executives*. National Institute of Sciences and Technology(NIST), Gaithersburg, MD, USA , 2012.

[262] Sztipanovits J., Bapty T., Karsai G. and Neema S. *Model-Integration and Cyber Physical Systems : A semantic perspective*. Institute for Software Integrated Systems, Vanderbilt University, TN, 2011.

[263] Sztipanovits J., Stankovic J. A. and Cornan D. E. *Industry-Academy Collaboration in Cyber-Physical Systems(CPS) Research*. White Paper, 2009.

[264] Taylor M., Diaz A. I., Sanchez L. A. J. and Mico R. J. V. *A Matrix Generalization of Dimensional Analysis: New Similarity Transforms to Address the Problem of Uniqueness. Advanced Studies Theoretical Physics*, 2(20):979–995, 2008.

[265] Tegmark M. *On the dimensionality of spacetime.* Class. Quantum Grav. 14 (1997) L69L75., 1997.

[266] The German Aerospace Center (DLR). *Railway Collision Avoidance System (RCAS) Project.* In *Avaliable at : http://www.collision-avoidance.org/rcas/*, 2016.

[267] Thornton R. *Integrating the Spatial Semantics of Verbs and Prepositions during Sentence Processing.* In *The Oxford Handbook of Cognitive Linguistics*, 2010.

[268] Tidwell D. *XSLT.* O'Reilly and Associates, Sebastopol, California, 2001.

[269] Tossebro E. and Nygard M. *Representing Uncertainty in Spatial Databases.* In *High Performance Computing and Simulation Conference, 141152*, 2008.

[270] Turing A. M. *On computable numbers, with application to the entscheiddungsproblem. Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.

[271] Unified Modeling Language (UML). See http://www.omg.org/uml. 2003.

[272] United States Government Accountability Office (US GAO). *Aviation Runway and Ramp Safety: Sustained Efforts to Address Leadership, Technology, and Other Challenges Needed to Reduce Accidents and Incidents.* Report GAO-08-29, United States Government Accountability Office, Report to Congressional Requesters, November, 2007.

[273] United States Government Accountability Office (US GAO). *Aviation Safety: FAA has Increased Efforts to Address Runway Incursions.* Report GAO-08- 1169T, United States Government Accountability Office(GAO), Testimony Before the Subcommittee on Aviation, Committee on Transportation and Infrastructure, House of Representatives, September, 2008.

[274] Varzi A. *Parts, Wholes, and Part-Whole Relations: The Prospects of Mereotopology.* Data and Knowledge Engineering 20(3), 259-286, 1996.

[275] Vieu L. *A Logical Framework for Reasoning about Space* . A. Frank and I. Campari (eds.): Spatial Information Theory - A Theoretical Basis for GIS, Proceedings of COSIT'93. Berlin, pp. 25-35, 1993.

[276] Vieu L. *Spatial representation and reasoning in Artificial intelligence* . Spatial and Temporal Reasoning, pp 5-41, Ed Springer, 1997.

[277] Wagner D.A, Bennett M. B., Karban R., Rouquette N., Jenkins S. and Ingham M. *An Ontology for State Analysis: Formalizing the Mapping to SysML* . IEEE Aerospace Conference, Big Sky, MT, USA, 2012.

[278] Wang Y., Ma J., and Knight B. *A Visualized Framework for Representing Uncertain and Incomplete Temporal Knowledge.* In *International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:7, No:11*, 2013.

[279] Wang Y., Yahya M., and Theobald M. *Time-aware Reasoning in Uncertain Knowledge Bases.* In *Fourth International VLDB Workshop on Management of Uncertain Data, Singapore*, 2010.

[280] Wassim N. G., Koopmann J., Smith J. D., and Brewer J. *Frequency of Target Crashes for IntelliDrive Safety Systems.* US Department of Transportation - National Highway Transportation Safety Administration, DOT HS 811 381, 2010.

[281] Weimer J., Bezzo N., Pajic M., Pappas G. J., Sokolsky O., and Lee I. *Resilient Parameter-Invariant Control with Application to Vehicle Cruise Control.* Control

of Cyber-Physical Systems: Lecture Notes in Control and Information Sciences, Volume 449, pp 197-216, 2013.

[282] Whitehead A. *Process and reality.* Macmillan, New York, NY, USA, 1929.

[283] Wilhelm R. and Grund D. *Computing takes time, but how much?* In *Communications of the ACM, Vol. 57, Issue. 2, February, 2014, pp. 94–103.*, 2014.

[284] Wing J. *Cyber-Physical Systems Research Challenges.* National Workshop on High-Confidence Automotive Cyber-Physical Systems, Troy, MI, USA , 2008.

[285] Winter D. *Cyber-Physical Systems - An Aerospace industry perspective.* Boeing Management Company, Seattle, WA, USA, 2008.

[286] Winter D. *Cyber-Physical Systems in Aerospace - Challenges and opportunities.* Safe & Secure Systems & Software Symposium (S5), Beavercreek, Ohio USA, June 14-16, 2011.

[287] Wolfgang M. *Leibniz's Theory of Space in the Correspondence with Clarke and the Existence of Vacuums* . Available at : http://www.bu.edu/wcp/Papers/Mode/ModeMalz.htm, Universitt Bonn, GE, 1995.

[288] Wong E. W., Debroy V. and Restrepo A. *The Role of Software in Recent Catastrophic Accidents.* EEE Reliability Society 2009 Annual Technology Report, 2009.

[289] Worboys M. F. *A unified Model for Spatial and Temporal Information. The Computer Journal*, 37:26–34, 1994.

[290] World Wide Web Consortium(W3C). *OWL 2 Web Ontology Language Profiles (Second Edition).* In *W3C Recommendation 11 December 2012, Available at: http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/*, 2012.

[291] Wuthishuwong C., and Traechtler A. *Vehicle to Infrastructure based Safe Trajectory Planning for Autonomous Intersection Management.* In *13th International Conference on ITS Telecommunications (ITST)*, 2013.

[292] XML Stylesheet Transformation Language (XSLT). See http://www.w3.org/Style/XSL. 2002.

[293] Ye Y., Lu H., Ma J. and Jia L. *Uncertain Temporal Knowledge Reasoning of Train Group Operation Based on Extended Fuzzy-Timing Petri Nets.* In *Advances in Soft Computing, vol. 40, pp 59-64*, 2007.

[294] Zeeger C.V., and Deen R.C. *Green-extension systems at high-speed intersections.* ITE Journal, 19 24, 1978.

[295] Zhang L., He J. and Yu W. *Challenges, Promising Solutions and Open Problems of Cyber-physical Systems. International Journal of Hybrid Information Technology*, 6:65–74, 2013.

[296] Zheni D., Frihida A., Ghezala H. B., and Claramunt C. *A Semantic Approach for the Modeling of Trajectories in Space and Time.* Advances in Conceptual Modeling - Challenging Perspectives, Lecture Notes in Computer Science Volume 5833, Springer(eds), pp 347-356, 2009.

[297] Zhou C., Ravn A. P., and Hansen M. R. *An Extended Duration Calculus for Hybrid Real-Time Systems.* Lecture Notes in Computer Science, Volume 736, pp 36-59, 1993.

[298] Zhou Y. and Baras J. S. *CPS Modeling Integration Hub and Design Space Exploration with Application to Microrobotics.* Control of Cyber-Physical Systems: Lecture Notes in Control and Information Sciences, Volume 449, pp 23-42, 2013.

[299] Zhu Q., Bushnell L., Tamer B. *Resilient Distributed Control of Multi-agent Cyber-Physical Systems.* Control of Cyber-Physical Systems: Lecture Notes in Control and Information Sciences, Volume 449, pp 301-316, 2013.

[300] Ziegler C. *Watch the moment a self-driving Google car sideswipes a bus.* In *The Verge, Available at: http* : *//www.theverge.com*/2016/3/9/11186072/*google* − *self* − *driving* − *car* − *bus* − *crash* − *video, accessed 10/03/16*, 2016.

[301] Zimmerman K. and Bonneson J. A. *Number of vehicles in the dilemma zone as a potential measure of intersection safety at high-speed signalized intersections.* 83rd Annual Meeting of the Transportation Research Board Washington, D.C., USA, 2004.

[302] Zimmermann K. *Without distances: the delta calculus.* A. Frank and W. Kuhn(eds.) Spatial Information Theory: A theoretical basis for GIS, COSIT'95, LNCS No. 988, Springer-Verlag, pp. 59-68, 1995.