Context and Interaction in the Internet of Things

Matthew Weber

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-114
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-114.html

August 15, 2019

Acknowledgement

This work was supported in part by the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The later
work in this thesis was supported in part by the National Science Foundation (NSF) award #CNS-1836601 (Reconciling Safety with the Internet) and the iCyPhy (Industrial Cyber-Physical Systems) Research Center supported by Avast, Camozzi Industries, DENSO International
America, Inc., Ford, Siemens, and Toyota. Special thanks to Lab 11 for sharing their SurePoint localization data for chapter 6.

Context and Interaction in the Internet of Things

by

Matthew E. Weber

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Edward A. Lee, Chair
Sanjit Seshia
Kenneth A. Bamberger

Summer 2019

Context and Interaction in the Internet of Things

Abstract

Context and Interaction in the Internet of Things

by

Matthew E. Weber

Doctor of Philosophy in Computer Science

University of California, Berkeley

Edward A. Lee, Chair

Future IoT killer apps leveraging ubiquitous sensors and actuators will create value in emergent properties of composition and contextual awareness. This thesis focuses on two key aspects of principled IoT design: (1) using contextual information from the physical world, and (2) enabling interaction and composition across distributed cyber physical systems. These challenges are mutually intertwined. Acquiring contextual information about the world (1) usually involves building applications to coordinate device interaction (2). Similarly, the decision an application makes about which sensors and actuators to interact with (2) should be governed by contextual information (1).

In this thesis I will present research in both directions: I propose semantic localization, a logical language as the interface between spatial modeling and spatial programming. I will present the accessors platform as a framework for building dynamic swarm applications, and give an architecture for connected car service discovery with semantic accessors. I will show how ontology matching adapters may be used to enhance distributed cyber physical system interaction. Finally, I will give an SMT solving algorithm for spatial ontology verification in sensor networks.

To my parents.

Contents

List of Figures

List of Tables

Acknowledgments

This work was supported in part by the TerraSwarm Research Center, one of six centers administered by the STARnet phase of the Focus Center Research Program (FCRP) a Semiconductor Research Corporation program sponsored by MARCO and DARPA. The later work in this thesis was supported in part by the National Science Foundation (NSF) award #CNS-1836601 (Reconciling Safety with the Internet) and the iCyPhy (Industrial Cyber-Physical Systems) Research Center supported by Avast, Camozzi Industries, DENSO International America, Inc., Ford, Siemens, and Toyota. Special thanks to Lab 11 for sharing their SurePoint localization data for chapter 6.

Chapter 1

Introduction

The task of integrating computing technology with the world around us has both awesome potentials and enormous challenges. Embedded computer systems were already ubiquitous in vehicles, microwave ovens, game systems, communication devices, grocery stores, military technology, and countless other domains before the rise of the Internet of Things (IoT) which has propelled the cyber and physical worlds even closer together. The IoT promises even greater possibilities straight out of science fiction: collaborative robot swarms capable of navigating dangerous environments where humans cannot safely go, ludicrously efficient industrial centers, wireless sensor networks able to improve agricultural yield or non-invasively scan the human brain, connected/autonomous cars which refuse to collide with other vehicles, green smart cities which nurture public health and protect citizens during a disaster, wearable health technology empowering an aging population to live independently, and virtual/augmented/mixed reality systems with the potential to fundamentally alter the human experience. The list goes on.

And yet these technologies have potential dark sides: A denial of service attack on a smart home could lock the residents out of their own refrigerator. Orwell's Nineteen Eighty-Four unsettled readers in 1949 with televisions that watched back, but in a household of Siris, smart appliances, and ubiquitous sensing, perhaps the television will be just the tip of the iceberg. If augmented reality is a dominant platform for human interaction, an identity thief might fool your real life acquaintances or turn a hacked display into a nightmare of disturbing images. Not to mention were he living in such an age, Henry David Thoreau might take issue with the spiritual consequences of such an artificial world. How will the world's economic systems cope with massive job loss resulting from fully automated factories and AI management? Is it possible for a legal system to fairly assign liability when an autonomous vehicle kills someone? How about when an autonomous weapon kills someone?

Engineers must acknowledge that technology does not exist in a vacuum, rather it shapes and is shaped by social phenomena [22]. I believe there is a profound arrogance to the belief that technology offers the sole solution to deep sociotechnical problems. Yet neither are system builders powerless in the face of overwhelming historical and market forces provided they respect their own limitations and follow principled design methodologies to create sys-

tems which adhere to well-defined behaviors in pursuit of the good ubiquitous computing has to offer.

1.1 Cyber-Physical Systems

In many ways the IoT is not a new field of study, but rather a rebranding of the academic discipline of Cyber-Physical Systems (CPSs). The term **Cyber-Physical System** was coined in 2006 by Helen Gill at the United States National Science Foundation to categorize *the class of systems which feature the interaction of cyber and physical components* [54]. As IoT devices can be described as CPSs with a network component, they also have a strong conceptual similarity to wireless sensor networks and networked robotics.

Principled engineering of CPSs relies heavily upon the concept of Model Based Design (MBD), wherein a system specification (i.e. a model) is used to formalize the description of a CPS [54]. The model guides implementation of the system and also facilitates verification so that the design may be analyzed for correctness. In some cases automatic synthesis methods may even be used to produce a correct-by-construction implementation directly from the model, eliminating the engineering work of implementation entirely. The MBD philosophy has produced massive historic successes such as electronic design automation, which revolutionized the integrated circuit industry and made chips with tens of billions of transistors possible from an engineering perspective [81].

CPSs can be modeled in many different ways at different levels of abstraction. For example, control theoretical models conceptually separate the physical system (plant) from its controller through sensing and actuation. Models of Computation may be threaded or message passing, concurrent or sequential, timed or untimed [79]. Hybrid Systems models describe the joint modeling of computation and physical dynamics via discrete and continuous behavior. As a result of learning, an artificial neural network builds a model expressed by its topology and weights. Models of the physical world might use quantum wave functions, relativistic models, or simple Newtonian physics. There is no simple choice for a "best" model in all situations because different CPSs have differing requirements. Even though a modeling technique like Newtonian physics might be flawed, it could still be the most appropriate choice for modeling a system of everyday objects.

In his book *Plato and the Nerd*, Prof. Edward Lee (my research advisor) identified a fundamental difference between the models used in science and in engineering [53]. Scientific models attempt to accurately represent the physical world. A scientific model is flawed when it fails to capture an important quality of the world. However, engineering models are specifications for things which may not yet exist in the physical world, and it is the fault of a *realization* of the model when the realization fails to agree with the model. In my view, principled design for the IoT requires both scientific and engineering modeling. It takes scientific models to learn about the state of the world, and engineering models to define the cyber, physical, and cyber-physical systems that act upon it. However, modeling for the

IoT is complicated by its architecture and the interaction of numerous systems which are by necessity designed with respect to different models.

1.2 IoT Architecture: Swarm, Fog, and Cloud

The term IoT is a catchall for a range of topics[1]. In this thesis I define the **IoT** as *a system of systems comprising the sensing and actuation of real-world "things" together with networked and embedded computational intelligence.* The standard architecture of the IoT has three main components: the Swarm, the Fog, and the Cloud.

The Swarm

The Swarm[2] refers to a *ubiquitous wireless networked collection of sensors and actuators embedded in the physical world.* Swarm devices are characterized by their sensing or actuation of physical world phenomena. A **Swarmlet** is *an application which governs the interaction of swarm devices.* Compared to the Cloud, the Swarm is a significantly more immature technology domain with unique engineering challenges. To reduce cost and/or power consumption, many swarm devices have constrained computation and networking capabilities. Example Swarm devices available today include smart watches, environmental sensors, and smart appliances. Future swarm devices may include brain-monitoring neural dust [84], mixed reality sensor/actuator networks, and advanced robotics.

The Fog

The Fog consists of *locally available networking and computation infrastructure within physical proximity of swarm deployments.* The name "Fog" is derived from both its conceptual position at the edge of the Cloud and the compromise it offers between Swarm and Cloud capabilities. Fog devices typically offer greater computational resources than Swarm devices (but less than the Cloud) and often serve as networking gateways to facilitate communication from Swarm to Cloud and back. Addressing the gateway problem makes Fog devices a key part of any IoT deployment: eg. a bluetooth device will be isolated from the internet without a Fog intermediary.

Due to the Fog's relative accessibility and proximity to the end user, Fog devices often provide the user interface for an IoT service. Example Fog devices include mobile phones,

[1] Other terminology includes the Internet of Everything, distinguished from the IoT by a focus on people and data; the Industrial Internet, concerned with industrial applications; Industry 4.0 [49] the next stage of industry following the industrial revolution (1.0), electricity (2.0), and widespread digitalization (3.0); and specific corporate visions like IBM's Smarter Planet

[2] The phrase "Swarm at the Edge of the Cloud" is credited to a keynote talk by Jan Rabaey at the VLSI Circuits Symposium in Kyoto June 15, 2011 [41]. The majority of the research in this thesis was funded by the TerraSwarm research center [55], a STARNet project for researching technology for a world with a trillion sensors.

edge gateway routers[3], and home voice assistants like Google Home or Amazon Echo. Future Fog devices may include connected cars, and quadcoptors capable of flying an internet connection to where it's needed.

While the gateway capabilities of the Fog are paramount, Fog computing can also offer application advantages over the cloud. Locally processed data is resilient to Cloud outages and protects privacy by simply not sharing information with the world at large. Fog computation has the added benefit of saving overall network bandwidth by avoiding unnecessary communication with the Cloud. While the Cloud usually has the upper hand in data processing throughput, in cases where the Fog's low (and more deterministic) latency has priority, improved performance can come from keeping computation local. In a quick test in my Berkeley apartment it took 4ms to ping my home router and 15ms to ping www.google.com, while global ping statistics[4] indicate international pings can exceed 300ms. For computationally light jobs with running time dominated by communication latency, Fog computation may be best for performance.

The Cloud

The Cloud is the popular name for *globally available computation resources centralized within powerful data centers*, and has already revolutionized modern computing. Many of the same advantages the Cloud brings to the conventional internet such as IT infrastructure as a service, globally available data storage, and high performance cluster computing are relevant to the IoT. However, there are unique ways the Cloud can act as the "brain" of the IoT. By aggregating Swarm data, worldwide patterns in sensing and actuation may be discerned and massive machine learning data sets regarding real world phenomena can be published for scientific and commercial benefit.

The IoT Cloud is subject to many of the same technical challenges facing non-IoT Big Data applications. Managing, storing, processing, or appropriately sampling huge data sets can be a difficult task. Furthermore, a tradeoff must sometimes be made between the efficacy of Cloud analysis and privacy. Promising techniques like summarization of image collections [94], differential privacy [21], and active privacy protection via insertion functions for opacity proprieties [103] offer solutions to some of these problems, but more work is needed to make the IoT Cloud an optimal resource for the Swarm.

1.3 Context and Interaction

This thesis focuses on two key aspects of principled IoT design: (1) using contextual information from the physical world, and (2) enabling interaction and composition across swarm

[3]While edge gateways are widely commercially available today, the TerraSwarm project (terraswarm.org) proposed the SwarmBox, a Bluetooth Low Energy (BLE) and Wi-Fi connected edge device in the form factor of a Wi-Fi router with a relatively powerful ASRock IMB-186-4300U motherboard.

[4]Courtesy of wondernetwork.com/pings.

applications. These challenges are mutually intertwined. Acquiring contextual information about the world (1) usually involves building a swarmlet that interacts with sensor devices (2). Similarly, the decision a swarmlet makes about which swarm devices to interact with (2) should be governed by contextual information (1). This thesis contains research in both directions.

I began this work with a special focus on location, one of the most important and challenging aspects of physical context. Location matters for the IoT in ways it does not for the Internet. There's a world of difference between illuminating a smart light bulb located in your home or one a thousand miles away. But while the physical location of a web server might affect the latency of communication or quality of service, it won't fundamentally change the content of the hosted page. For a Swarm device, its physical relationship with the world has everything to do with what it can and cannot accomplish.

The capabilities of a swarm device are similarly enhanced (or limited) by its interaction with the rest of the IoT. On its own a swarm device is restricted to whatever sensors and actuators it has built in. When composed with other devices, it can be re-purposed for applications its original designers did not anticipate into a whole greater than the sum of its parts. Even on a theoretical basis, interaction is more powerful than algorithms [101].

I believe future IoT killer apps will create value in *emergent properties of composition and contextual awareness.* Most IoT systems today are **static IoT services** *designed for the specific sensors and actuators of a particular device.* The goal of this research is to raise the level of abstraction for IoT design, so that contextual specifications for IoT interaction (e.g. "whenever I'm inside a room, turn on the lights there") may be automatically converted into a working swarmlet made up of arbitrary hardware communicating over arbitrary networks. Such a **dynamic IoT service** *opportunistically discovers contextually relevant devices and combines them into new IoT applications.*

1.4 IoT Privacy and Security

Many of today's consumer IoT devices today seem to be built by slapping (buggy) sensors and microcontrollers onto standalone objects that have no business being digitized. Some of the resulting products are chronicled by the popular twitter account "Internet of Shit"[5] and include an internet connected block of wood, a YouTube connected smart sofa, and smart sneakers which "'can't be tightened or properly worn' when there's a bug in an update"'. These outcomes might have been avoided by focusing on contextually appropriate interactivity that does not interfere with the main functionality of the object.

Perhaps unsurprisingly, IoT devices are known for their security vulnerabilities. IoT development often requires expertise in both web and hardware domains, and it seems the engineers behind IoT devices often lag behind the best practices for secure web design. In 2016 the Mirai botnet exploited this weakness by repurposing IoT devices with default password settings to create some of the most powerful denial of service attacks in history

[5] twitter.com/internetofshit

[46]. It is unacceptable for physical devices capable of sensing and actuating within our own homes to have *worse* security than web apps, yet that is often the state of the world in 2019.

Many in the tech industry view privacy as a lost cause, and to me this is equally unacceptable. IoT devices with unprecedented visibility into the most personal aspects of our day-to-day lives demand "no-spying" guarantees. Historically, the United States has followed a legal philosophy toward privacy in which users may opt out of data collection by refusing to use a service. This approach may not work in a future with ubiquitous sensing where opting out is impossible. The EU's General Data Protection Regulation (GDPR) is a recent law concerning limits on how companies may process their user's personal data. While GDPR is a step in the right direction for web technologies, it needs significant fine tuning to be applicable to IoT devices. For example the mechanism through which consent for data collection from devices without screens (eg. smart lightbulbs) will be obtained is currently very ambiguous.

Laws like GDPR are usually concerned with how companies collect, store, and share user data. Engineers (particularly academics) tend to view privacy differently, in terms of whether or not it is theoretically possible to obtain someone's personal data. For example, if user-encrypted data is stored in a corporate database, the legality of selling data to advertisers is irrelevant: it can't be done unless the advertises have somehow stolen the user's encryption key. Such approaches to privacy are known as privacy by design [80] and are an essential complement to legal privacy for the IoT. Privacy by design may be as sophisticated as homomorphic encryption, or as simple as splitting up the rows of a database to separate personally identifiable information from data concerned with a user's day-to-day usage of the service.

Ultimately, I believe MBD is a key approach to creating IoT systems with both security and privacy. The modeling step forces engineers to consider failure modes/security vulnerability of the system and to determine which data is important and worth collecting. Privacy by design is a natural byproduct of explicitly modeling information flow. While it is often difficult for engineers to build secure systems from scratch, using a library of correctly implemented secure networking components is much harder to get wrong (chapters 3 and 4). Chapters 2 and 6 address consistency checking for security in spatial models, and a mechanism useful for splitting up spatial/contextual models for improved privacy is given in chapter 5.

1.5 Overview

The contents of this thesis are organized around modeling, design, and analysis for engineering context-aware and interactive swarmlets. Chapter 2 begins with semantic localization, a unifying formalism for describing (physical) location in the Swarm. Chapter 3 introduces the accessor platform, a (cyber) programming model for composing swarm devices into integrated applications. Chapter 4 shows how accessors can be combined with contextual information about the world for semantic service discovery with connected cars as a motivating exam-

ple. Chapter 5 demonstrates the use of ontology matching components called "adapters" for building dynamic IoT services. Finally, chapter 6 presents an algorithm for error and attack detection in a sensor network ontology as an applied example of inference and consistency checking in spatial ontologies. The narrative across chapters has a cyclic structure: In early chapters accessors are used to obtain information for semantic localization. In later chapters semantic localization is used to obtain accessors.

A common philosophical thread linking these approaches to context and interaction is an aversion to proposing the universal adoption of IoT standards. When applied to narrow and well-defined domains, standardization can often be the easiest way to ensure effective coordination across implementations. But I believe most of the world is far too messy, disorganized, and diverse to perfectly fit a single networking API or contextual model. Not only would universal adoption of such a standard be impractical, it would not take long before someone invented something new and incompatible (for example, too low power to use the networking API). Together, semantic localization (Chapter 2) and Accessors (Chapter 3) are designed to bridge silos in contextual modeling and IoT communication respectively.[6]

Collaborators

I am indebted to many people for their involvement in the research presented in this thesis. From here on I will be switching to the pronoun "we" to refer to myself and the collaborators/coauthors listed below for each chapter.

- Chapter 2. This chapter is adapted from unpublished work coauthored by Edward A. Lee.
- Chapter 3. This chapter is adapted from our paper "A Component Architecture for the Internet of Things" [15]. Collaborators in this work include Christopher Brooks, Chadlia Jerad, Hokeun Kim, Edward A. Lee, Marten Lohstroh, Victor Nouvellet, Beth Osyk, and many others working on the accessors project.
- Chapter 4. This chapter is adapted from our paper "Service Discovery for The Connected Car with Semantic Accessors" [98], coauthored by Ravi Akella and Edward A. Lee.
- Chapter 5. Collaborators on the ideas in this chapter include Chadlia Jerad and Edward A. Lee.
- Chapter 6. This chapter is adapted from our paper "Gordian: Formal Reasoning Based Outlier Detection for Secure Localization" [100], coauthored by Baihong Jin, Gil Lederman, Yasser Shoukry, Edward A. Lee, Sanjit Seshia, and Alberto Sangiovanni-Vincentelli.

[6] Arguably Accessors are themselves standards in a meta sense in that they have definitions and may be incompatible with other standard-bridging schemes, but this is an unavoidable concern risking infinite regression.

Chapter 2

Spatial Ontologies and Semantic Localization

Today, we have mature theories of computation, developed over the last 80 years or so, and mature theories of physical structure and dynamics, developed over the last 300 years or so. But we have only the barest beginnings of theories that conjoin the two. One of the key points of friction is that the notion of location in space and time are central to a physical reality, but absent in a cyber reality. When the focus is mutual imitation, as in simulation, it is natural to construct cyber representations of space and time by approximating positions in a Euclidean geometry and Newtonian time with floating point numbers. But when the goal is cyber-physical collaboration, such a literal representation of space and time may not be the best choice.

When considering mobile devices and the IoT, applications often care more about logical spatial and temporal relationships than quantitative ones. To preserve security and privacy, for example, one device may be granted access to data held by another device only when the two devices are in the same room at the same time. The notion of "same room at the same time" is an example of what we call *semantic localization*. It is not so much about geometric location, but rather asserts a "semantic" spatial relationship.

In this chapter we survey location models from robotics, the internet, CPSs, and philosophy. The diversity in these models is justified by differing application demands and conceptualizations of space (spatial ontologies). To facilitate interoperability of spatial knowledge across representations, we propose a logical framework wherein a spatial ontology is defined as a model-theoretic structure. The logic language induced from a collection of such structures may be used to formally describe location in the IoT via semantic localization. We finish the chapter with definitions for open ontologies and logical inference.

2.1 Location as Context for Cyber-Physical Systems

For all its importance to understanding cyber-physical systems, localization, the challenge of determining the location of physical objects, remains an open problem. GPS, which has been a resounding success for outdoor localization, relies on direct line-of-sight signals from satellites, and is consequently ineffective for indoor environments or outdoor environments where obstructions such as buildings interfere with measurements. Researchers have been trying to address the indoor localization problem since the early 1990s with systems like Active Badges [97] and Cricket [78], and yet even to this day, a general purpose, accurate, cost effective, deployable system with the potential to reach the ubiquity of outdoor GPS remains elusive. A big part of what makes the problem difficult is the potential for interference in indoor environments where walls, furniture, and people, obstruct and reflect signals. Even something as simple as turning on a microwave oven causes interference to RF signals and might disrupt signal strength measurements for an indoor localization system operating in the 802.11 bands. Nevertheless, we are optimistic that in the near future, applications will routinely have available a variety of types of location information with a range of quality. This chapter addresses how to organize and use that location information.

The most commonly articulated purpose for indoor positioning is indoor navigation. There is no doubt a market for apps that can help you find your way in whatever building you happen to be inside, but in our view this is probably a small market that dramatically understates the potential of contextual awareness in the IoT. The future of indoor and outdoor space-aware cyber-physical systems involves scenarios where position in space is less important than spatial interrelationships. Consider a fleet of self-driving cars, where proximity in driving time, energy, and ride sharing opportunities are more useful criteria for control than geo-coordinates. Indoors, having awareness of which devices are in the same room may be more useful than measurements of their position in two or three-dimensional space. For such applications, different representations of space than a coordinate-system based physical map become appealing.

Relational ontologies of space aren't wildly foreign concepts; they can be found in some of today's apps. Take data from FourSquare, the app that lets users "check into" locations, as an example of a non-geometric representation of space. A user checked into a restaurant on FourSquare is known to be inside the establishment, but it would be a mistake to guess exact geocoordinates for him/her and plot them inside the restaurant's perimeter because they might be sitting at a table or standing by the door, and precise geocoordinates would suggest a false confidence as to the nature of unknown information. Unplotability doesn't make the FourSquare data somehow less accurate or reliable than a physical coordinate map, it just makes it different. We call this kind of *geometrically fuzzy yet logically precise spatial information* **semantic localization**.

Designing A Robo-Cafe

In collaboration with researchers at U Penn, Michigan, UW, CMU, and Berkeley, in 2015 we demonstrated a robotic delivery system at the DARPA "Wait, What?" conference where users could place an order on a smart phone localized by the ALPS Ultrasound Localization System [51] and have a desired snack delivered to their location by a roaming Scarab Robot [67]. The demo was designed to showcase integration and composability of cyber-physical systems via accessors (see Chapter 3), but most relevant to this chapter is the spatial interaction needed between the Scarab and ALPS.

The Scarab comes equipped with a laser rangefinder which it uses with standard ROS packages to perform Simultaneous Localization and Mapping (SLAM) and to build an occupancy-grid map of its environment (see Figure 2.1). An occupancy grid is a fairly simple data structure commonly used in robotics to represent an environment (modeled as a grid over 2D or 3D Euclidean space) that is essentially a big array with values from 0 to 100. A value of 0 indicates the robot is almost certain the cell does not contain an obstacle, and a value of 100 that the cell is almost certainly impassable. The robot also maintains an estimate of its pose (position and orientation) at the cell where it is currently located.

The second localization system, ALPS, uses ultrasonic beacons, and is also deployed in the DOP Center (Figure 2.1). The system is deployed by placing beacons at known locations in a building and finding the correspondence between the beacons and coordinates on the building's floor plan. The beacons send time synchronized chirps of ultrasound in the 20kHz to 22kHz bands that are beyond the range of human hearing but receivable at the standard sampling rate of a cell phone microphone. A smartphone with an ALPS app can locate itself on the floor plan's coordinate system.

When the robot is localized on its occupancy grid and the phone is localized on the floor plan, the Scarab uses ROS navigation packages to deliver a snack. However, there is a rather subtle challenge in the last step: the phone has known coordinates on the floor plan and the robot is at a known cell of the occupancy grid, but the two are, as given, totally unrelated! Deployment of the Robo-cafe requires a coordinate system alignment phase in which ALPS's model of space is brought into concordance with the Scarab's model, but this is a one-off solution to a very general problem that we introduce in this chapter and fully explore in chapter 5.

Spatial Ontologies

Any cyber-physical system that seeks to interact with the physical world assumes a model of space, either explicitly or implicitly. Such a model is a spatial ontology. Broadly, the subject of ontology from philosophy is a study of the nature of existence, what it means for something to be and to be something. In computer science, ontology is usually about association of entities in a model with structured taxonomies, addressing questions like "is this object an instance or example of that class of objects?" In prior work, it has been shown that useful ontologies can be constrained to have a mathematical lattice structure, and that

Figure 2.1: Occupancy grid formed by a Scarab robot roving the DOP Center at Berkeley. This image shows use of a relatively poor distance sensor on the roving robot, measuring for example received signal strength from another object, and then applying a particle filtering algorithm constrained by the occupancy grid map to estimate the position of the other object. The red dots are the particles, the green square is the target, the blue square is the Scarab robot, and the black areas are occupied grid points as detected by the lidar rangefinder on the Scarab. The grey areas indicate where the occupancy grid has no information. Image courtesy of Ilge Akkaya.

they thereby acquire enormous algorithmic and formal benefits that can be leveraged to compose ontologies, perform inference, and check correctness [59, 58, 57]. Such ontologies form a subset of commonly used ontology frameworks such as OWL (discussed at greater length in Chapter 4). Their mathematical structure resembles that of Hindley-Milner type systems, from which they inherit practical algorithms that scale to very large numbers of elements. For example, type inference maps into the problem of finding a fixed point of monotonic function over a lattice.

In computer science, an ontology is a collection of *concepts* and relations between those concepts. The most commonly used relations represent "is a" (taxonomy) and "part of" (meronymy) relationships. The coordinate system alignment problem in Robo-Cafe is in fact an instance of a general problem that must be addressed whenever two cyber-physical systems seek to work across contextual ontologies. A central motivation for the modeling framework presented in this chapter is to formalize the structure of spatial ontologies for the development of mappings and relationships that enable heterogeneous mixtures of ontologies in cyber-physical applications. This modeling framework motivates the use of Semantic Web technologies in Chapter 4 for connected cars and motivates the algorithmic approach of

Chapter 5.

Spatial ontologies have more diversity than just choice of coordinate system. A common dichotomy in ontologies is the distinction between "objects" and "fields" [91, 33, 26]. An "object" is an entity that is distinct, with a clear boundary, and in the language of [26] is "individual and fully deniable." Examples of objects include: an apple, a table, or a flashlight. A "field" describes phenomena without clearly defined boundaries that are "smooth, continuous and spatially varying" [26]. The magnetic field emanating from a hand-held bar magnet is a good example of this concept. From a certain pedantic perspective the field is present everywhere in the universe, only its strength is almost everywhere so weak as to be negligible. Some geographical features like lakes have elements of both objects and fields because it can be hard to identify where they end.

Spatial ontologies can also vary with respect to their interpretation of entities with respect to time. SNAP and SPAN are two cooperative ontologies proposed by Grenon and Smith [33] to capture the distinction between "continuants," objects with an identity that persists across time, and "occurants," processes defined in part by their beginning and ending. Examples of continuants include the planet earth or a pair of shoes because it makes sense to consider their spatial properties at a particular snapshot of time. The same is not true for occurants like a volcanic eruption or the takeoff of a helicopter. Such occurants unquestionably have a spatial existence but their reality is best comprehended in four full dimensions; a sequence of 3D observations misses something essential about the nature of the process. There is clearly a strong interrelation between SNAP and SPAN ontologies. This point is not missed by Grenon and Smith, who devote a latter section of their paper to trans-ontology interrelations between SNAP and SPAN.

Any discussion of existing ontologies on the web would be remiss without mentioning the Semantic Web. The Semantic web is a W3C effort first started in the early 2000s to modify ordinary HTML webpages with special markup to label their semantic content. The hope is that when markup is combined with a collection of ontologies for web content and real-world objects, algorithms will be able to apply ontological reasoning to web elements and data. For example, an image of a bridge embedded in a web site could be labeled as such and found through a general search for "landmarks" by using the ontological information that a bridge *is* a landmark. According to the wikipedia article on the semantic web, by 2013 some 4 million web pages had been augmented with semantic web information. But this is done primarily through human intervention, which could account for the relatively modest penetration compared to the total number of web pages.

Standards for Spatial Representation

Many standards for spatial representation have been proposed in different domains, a sample of which is presented here.

According to Lieberman et al., as of 2007 the semantic web maintained at least seven varieties of spatial ontologies [60]. These include Geospatial Features, Feature Types, Toponyms / Placenames, (Geo) Spatial Relationships, Coordinate Reference Systems, Geospa-

tial Metadata, and (Geo) Web Services. The relationships between these, however, are highly unstructured and lacking in formal properties that can exploited algorithmically.

A popular spatial ontology today is codified in a JSON schema called GeoJSON [16]. This is used by many location based services. In contrast to the semantic web, GeoJSON is good at representing geometries, but not higher level ontological concepts and relationships. It supports points, lines, polygons, and collections of polygons in 2D or 3D. Given the extensive support for GeoJSON in existing apps and software, it is a useful standard to leverage for geometric concepts. But restricting spatial ontologies to exclusively geometric concepts is a mistake. Spatial relationships are more complex.

On the opposite end of the complexity spectrum, the OpenGIS Geography Markup Language (GML) Encoding Standard [77] is a 437 page specification document describing an XML schema for spatio-temporal ontologies. It follows the ISO 19101 definition of a feature as an "abstraction for real world phenomena" and represents the world as a collection of features defined as name, type, value triples. The increased complexity allows for the description of more sophisticated data such as spatial geometries, spatial topologies, time, coverages, and observations. The format can be extended to application schema such as IndoorGML [56] which is targeted for indoor navigation. IndoorGML focuses on layered graph representations of relationships such as adjacency and paths between semantic objects in indoor space. It models the world as a collection of cells representing geometry and topology via the Poincaré duality to achieve a "Multi-Layered Representation" of a given space in different contexts.

A variety of geometric data structures and algorithms are employed in the field of computational geometry when high performance is desired for spatial analysis [10]. For example, a doubly-connected edge list is used for the thematic map overlay problem, in which the overlay of spatial subdivisions is computed.[1] A trapezoidal map is another geometric data structure employed to solve point location queries: given the coordinates of a point and a map subdividing the plane into regions, determine which region contains the point.

2.2 Location Modeling

The purpose of location modeling in this work is to support a logic for reasoning about spatial ontologies across independently designed cyber-physical systems. By moving beyond just geometric position, this logic offers the possibility for a much richer set of applications than just navigation, including for example security (e.g. restricting access to some service to only devices in the same room); asset tracking (e.g. where is the remote control for this device, or the device for this remote); spatial search (e.g. find a temperature sensor in the same room as a mobile device); commissioning (e.g. deploying sensors and actuators without manually specifying their location); and context-aware services (e.g. lighting systems that automatically adjust to usage patterns of a room). We explore the use of semantic repositories to enable such applications in Chapters 4 and 5 but beyond this thesis there is

[1] Imagine overlaying two circles to form a venn-diagram, but with polygons instead of circles.

room for a larger suite of software components and services for creative application designers to use when reasoning about location information. Such services could handle mobility (e.g. notification when a device is no longer in the same room) and superposition of disjoint maps constructed at different semantic and geometric layers (e.g., relating geometric information to "in the same room" semantic information).

Model Theory

Model theory is a domain of mathematical logic originally developed to analyze logical formulas regarding mathematical structures such as groups, graphs, and fields. The key observation behind model theory is that logical formulas can be written to express properties in a manner independent of the mathematical structures with respect to which they are evaluated. For example the formula $\exists n\ 0 < n < 1$ is true with respect to $\mathbb{R}$ or $\mathbb{Q}$, but not with respect to $\mathbb{Z}$ or $\mathbb{N}$. A model (or structure) specifies a domain, such as $\mathbb{R}$, and gives interpretations to the symbols 0, 1, and $<$ so that their particular relationship may be determined. We summarize the fundamentals of model theory relevant to CPS location modeling below. The main reference for the following definitions is [102], which may be referred to for a more comprehensive introduction to model theory.[2]

A **formula** is a logical statement constructed in the usual way from:

- logical symbols: $\rightarrow$, $\leftrightarrow$, $\neg$, $\wedge$, $\vee$, $\forall$, $\exists$,), and (
- variables (a countably infinite collection)
- function symbols (eg. $+$ for a group operation)
- relation symbols (eg. $\leq$ for the ordering relation on $\mathbb{R}$)
- the relation symbol $=$, as the usual "equal sign"
- constant symbols which represent a particular element from the domain (eg. 0 or π)

The arity of function and relation symbols is ≥ 1.

A **signature** is a particular set of function, relation, and constant symbols. The **language** of a signature is the set of well-formed formula expressible using functions, relations and constants from the signature. A variable v_0 is **bound** iff it appears in a subformula (i.e. a syntactically correct part of a formula) following $(\forall v_0)$ or $(\exists v_0)$. Otherwise the variable is **free**, and may be assigned a value separately. For example: formula ϕ with free variables $v_0, v_1, ..., v_k$ may be written as $\phi(a_0, a_1, ..., a_k)$ to express the assignment of a_0 to v_0, a_1 to v_1, and so on.

The **sentences** of a language are formula of the language with no unbound variables. A **structure** (or model) is a tuple $\mathbb{A} = (\mathbf{A}, I)$ where $\mathbf{A}$ is a domain, i.e. a non-empty set, and I is an interpretation function. I maps function, relation, and constant symbols

[2] A friendlier introduction can be found at https://plato.stanford.edu/entries/modeltheory-fo/

to functions defined over $\mathbf{A}$, relations defined over $\mathbf{A}$, and elements of $\mathbf{A}$ respectively. A structure $\mathbb{A}$ **models** a sentence S of a language when the interpretation of the sentence within the structure evaluates to true. This relationship is denoted by $\mathbb{A} \models S$. and its converse by $\mathbb{A} \not\models S$.

A language with a finite signature may be concisely written as a tuple, eg. $\mathbb{L} = \{<, 0, 1\}$. Similarly, a model's domain and interpretation for that finite signature may be informally written as an analogous tuple, eg. $\mathbb{A} = (\mathbb{R}, <, \mathbf{0}, \mathbf{1})$. Here, $\mathbb{A}$ is the structure with domain $\mathbb{R}$ which interprets $\mathbb{L}$ with the strict ordering relation $<$, and constants 0 and 1.

Putting it all together, we may now formalize the motivating observation from the beginning of this section that the same formula may be true or false with respect to different domains. Regarding the example formula $\exists n\ 0 < n < 1$ we have $\mathbb{A} \models \exists n\ 0 < n < 1$, but for $\mathbb{B} = (\mathbb{N}, <, \mathbf{0}, \mathbf{1})$, $\mathbb{B} \not\models \exists n\ 0 < n < 1$.

Semantic Localization

We propose using the concepts of model theory to formally describe location in swarm systems. A spatial ontology can be represented as a structure $\mathbb{A} = (A, I)$. For $\mathbb{A}$ to be useful as a model of the space, most likely the elements of A should be places or things located at places. Similarly, I should provide spatially meaningful interpretations of relations, functions, and constants. The language for such a structure, will then consist of semantic localization statements.

Defining semantic localization as a model-theoretic language has the advantage of separating the specification of spatial reasoning from its implementation within a particular spatial ontology. Just as the formula $\exists n\ 0 < n < 1$ may be evaluated within different structures, so too might a semantic localization formula be evaluated within heterogeneous spatial ontologies. For example, let $contains(a, b)$ be a binary relation which is true when room a contains person b, let $user1$ and $user2$ be constants for people, and let the variable $room$ range over a set of rooms. The formula $\exists\, room\ contains(room, user1) \wedge contains(room, user2)$ expresses the spatial arrangement in which $user1$ and $user2$ are both within the same room, independently of a particular spatial ontology. We propose using semantic localization as a conceptual interface between location programming and location models in the swarm.

A semantic localization formula can be interpreted in one of two ways: either as an event condition or as a query into some spatial database. In the first case, the sentence acts as a predicate that triggers an event when it evaluates to true. In the second case, the formula can be evaluated against database entries to signify that the entries to return are those that cause the formula to evaluate to true when plugged into unbound variables. However, in either case a statement can only be evaluated in an ontology with a compatible signature.

Figure 2.2 represents a central idea governing location modeling, relating mathematical structures to the spatial connectives (relations) of physical objects in a CPS which they are capable of evaluating. Applying the concepts from model theory to CPS location modeling has the added advantage of enabling mathematical analysis to bring the theorems of model

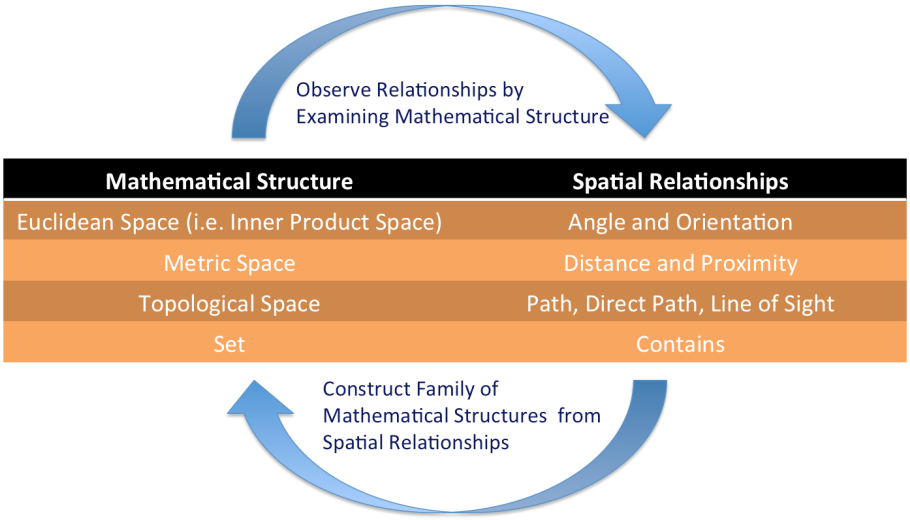

Figure 2.2: A comparison between mathematical structures and corresponding evaluable spatial relationships as described in our previous work [99].

theory to bear on the relationships spatial ontologies have to one another, such as embeddings (see chapter 5).

As suggested in Figure 2.2, spatial ontologies may be used to reason about spatial connectives, or spatial connectives may be discovered by sensors and used to construct mathematical structures.[3] Relations represent the structural aspects of the space (e.g. containment, path, proximity, angle, etc.), and functions define other structural aspects of the space (such as distance for a metric space) as appropriate. The values of constants, relations and functions are potentially time varying as the structure evolves. For instance, a topological ontology of an indoor space with doors opening and closing has a dynamic "path" relation. The quality and nature of the sensor data may constrain the level at which these ontologies may be constructed. For example, orientation information may simply not be available.

Physical and Relational Ontologies

In the previous section, a spatial ontology is a mathematical structure which can be used to evaluate a logical sentence that makes reference to spatial relationships. This notion of a spatial ontology is considerably more general than the usual notion of a printed paper map with a 2D representation of the road network of a city, for example. We will use the term "relational ontology" when we want to emphasize the abstracted nature of the spatial relationships that the map represents, but in this research, a spatial ontology is a mathematical object at any of these levels of abstraction, as long as it encodes some form of

[3] As in chapter 6 where a collection of internode distance measurements is used to construct a 2D Euclidan space representation.

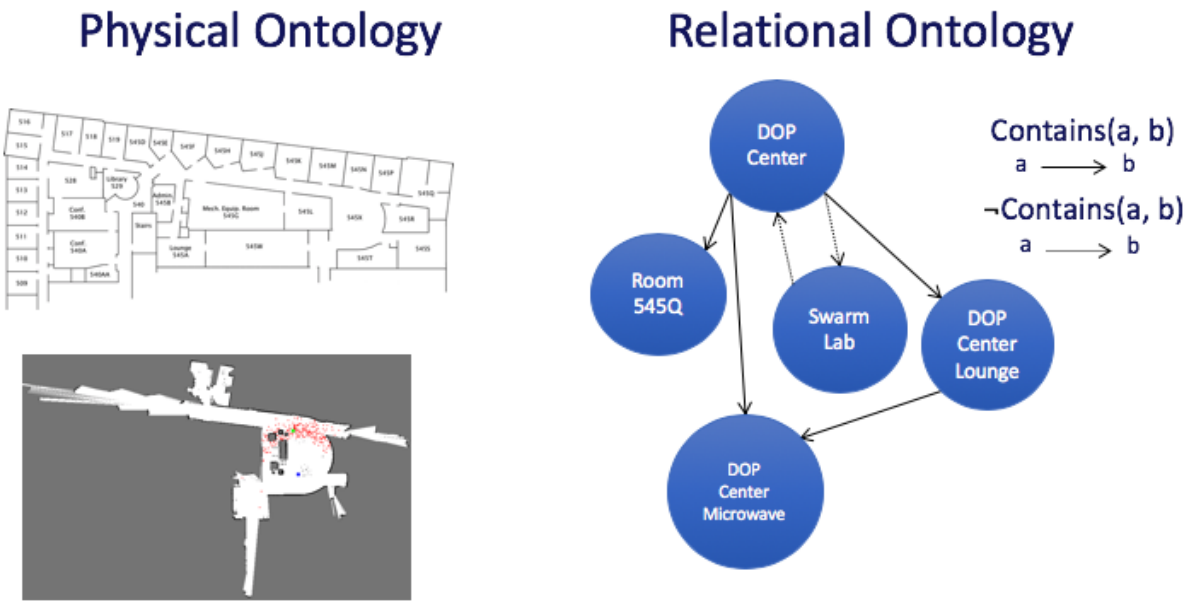

Figure 2.3: Concrete examples of a Euclidean-space ontologies vs. an abstracted relational ontology that represents only containment relations.

spatial relationships. For example, Figure 2.3 shows a relational ontology that is a partial order induced by the containment relation between sets; the relational ontology does not say anything at all about geometric properties such as distance or orientation.

As they get more abstract, of course, relational ontologies lose the ability to evaluate some kinds of spatial relationships. This idea parallels the usual hierarchy of mathematical spaces. A Euclidean space has quite a lot of mathematical structure that may not match well with the information available sensors are able to deliver. A Euclidean-space ontology supports reasoning about angles and orientation, concepts that are not defined in the more abstract mathematical structures shown in Figure 2.2.

Space-aware services should be constructed for the signatures of the most abstract spatial ontologies as possible. This will enable them to operate in more sensor-poor environments, to benefit from a greater variety of sources of spatial information, and to better preserve privacy by not handling information that is not needed. For example, the set containment relation is all the map information necessary for the FourSquare localization example in the introduction, since the only information to be gained from checking in is containment.

The hierarchy of these mathematical spaces offers a starting point for reasoning about combinations of maps. For example, given a Euclidean-space map of an office space and a Set (containment) map of objects in the space, objects can be placed approximately, with known error bounds, onto the Euclidean-space map. But much more complicated mapping combinations will be required, since even two Euclidean-space maps may not use the same coordinate system. The concept of a spatial ontology becomes an essential feature of location

modeling.

Topological spaces can be used to construct maps that represent paths through indoor settings. Navigation with graphs is a common concept in robotics [47], where nodes represent waypoints in a space and edges represent paths between waypoints. Such data structures are routinely used to construct sequences of actions to move a robot between nodes. Additionally, Ghrist et al. [29] show that algebraic topology can be used directly to relate the convex hull of a landmark set in a Euclidean space to a simplex of a simplicial complex. This provides a natural abstraction mechanism for topological maps.

Non-Euclidean metric maps are useful when the standard Euclidean metric does not really capture the interesting properties of a space. Consider a point x on the third floor of a building and the point y directly below it on the second floor. Points x and y are very close to each other in Euclidean space, but for the purposes of navigation, this misrepresents reality. We can instead define a metric space with metric D, where

$$D(x,y) = \begin{cases} \text{minimum length a continuous path from } x \text{ to } y, \text{ if there is such a path} \\ \infty, \text{ otherwise} \end{cases}$$

This is easily shown to be a metric (or even an ultrametric, for some graph-structured metric spaces). If stairways and elevators are not navigable open space for a particular robot, then this metric will yield $D(x,y) = \infty$, considerably more than the Euclidean distance.

An inner product space (of which a Euclidean space is a common variety) introduces the notion of angles. Angles can facilitate special kinds of analysis like trilateration, and the use of trigonometric angle measurements to localize objects in coordinate space.

As these examples illustrate, there are practical reasons to construct non-Euclidean ontologies. However each of these mathematical ontologies has the property that any map entity placed at a particular coordinate takes on all spatial relationships to other map coordinates implied by the structure of the space. This is undesirable when only a portion of those relations are positively known to be true and the rest are unknown. A key advantage of relational ontologies is the expression of *open ontologies*, where the absence of a relation does not imply its converse. This is analogous to ancient maps that provided useful navigation information despite significant distortions in the geometry and large gaps labeled "*terra incognita*". Open ontologies translate naturally into action plans that can deal with incomplete information.

This increased flexibility comes at the cost of a slightly more verbose vocabulary for relations. Consider the containment map on the right hand side of figure 2.3. Because this ontology is open, knowing that one place is not contained by another isn't enough to know they have no space in common. Another relation, "disjoint," is necessary to express that positive fact explicitly. We hope that the reader can see here a connection here to intuitionisitic logic, in that for open ontologies it is not enough to know a spatial relation is not not true to infer that it is true. Instead, relations must be constructively built up from known facts.

Formalizing Open Ontologies

An open ontology, $\mathbb{A}$ is a way of expressing partial knowledge about a spatial structure. If we take the philosophical position that the unexpressed information in an open ontology is fundamentally unknowabable, there is nothing to be done to increase the amount of information represented in $\mathbb{A}$. However, if we assume the missing information is knowable and could be expressed in an idealized (but hypothetical[4]) ontology $\mathbb{A}^*$, we may consider logical inference as a means to obtain information available in $\mathbb{A}^*$ but not $\mathbb{A}$.

We formalize this notion below, but first some definitions. Let $\perp$ be the symbol for "unknown".[5]

Definition 2.2.1 (Partial Order on Functions)**.** *We define a (pointwise) partial order on n-valued function $f : \boldsymbol{A}^n \to (\boldsymbol{A} \cup \perp)$ with $f \leq f'$ iff for $x \in \boldsymbol{A}$, $f(a_1, a_2, ...a_n) = x \to f'(a_1, a_2, ...a_n) = x$.*

Observe this definition allows $f(a_1, a_2, ...a_n) = \perp$ with $f'(a_1, a_2, ...a_n) = x$. In other words, f agrees with f' everywhere where f is not unknown, but may disagree where f is unknown.

Let open ontology $\mathbb{A}$ and its idealized $\mathbb{A}^*$ both be structures with the same signature and the same domain. $\mathbb{A}$ may be missing some information available in $\mathbb{A}^*$.

Definition 2.2.2 (Partial Order on Open Ontologies)**.** *We define a pointwise partial order on open ontologies $\mathbb{A}$ and $\mathbb{A}^*$ with the same signature and domain ($\boldsymbol{A}$) by ordering relation $\sqsubseteq$. The $\sqsubseteq$ relation indicates $\mathbb{A}^*$ has more information than $\mathbb{A}$ when:*

- $\mathbb{A}$'s functions may have unknown value ($\perp$) over some elements of the domain where $\mathbb{A}^*$'s functions are known. With $f_{\mathbb{A}}$ as the interpretation of function symbol f in $\mathbb{A}$ and $f_{\mathbb{A}^*}$ as the interpretation of f in $\mathbb{A}^*$, $f_{\mathbb{A}} \leq f_{\mathbb{A}^*}$.
- $\mathbb{A}$'s relations may be missing tuples which are available in the analogous relations of $\mathbb{A}^*$. For example, with $r_{\mathbb{A}}$ and $r_{\mathbb{A}^*}$ as interpretations of relation symbol r in models $\mathbb{A}$ and $\mathbb{A}^*$ respectively, $r_{\mathbb{A}} \subseteq r_{\mathbb{A}^*}$.
- $\mathbb{A}$'s interpretation of constant symbols may be less complete than the interpretation of $\mathbb{A}^*$. With k as the set of constant symbols in $\mathbb{A}$'s signature and $c : k \to (\mathbf{A} \cup \{\perp\})$, as the function mapping constant symbols to domain elements, $c_{\mathbb{A}} \leq c_{\mathbb{A}^*}$.

Not only does the ordering relation defined by $\sqsubseteq$ relate $\mathbb{A}$ to $\mathbb{A}^*$, it also relates $\mathbb{A}$ to a chain of non-idealized open ontologies $\mathbb{A} \sqsubseteq \mathbb{A}' \sqsubseteq \mathbb{A}'' \sqsubseteq ... \sqsubseteq \mathbb{A}^*$ with progressively more

[4]Of course we don't actually know the contents of $\mathbb{A}^*$ because it contains the information we currently don't know in $\mathbb{A}$. But as we will discuss here and in section 5.2, it is nevertheless useful to define $\mathbb{A}^*$ as a model so we may make explicit our assumptions about the missing information.

[5]We do not always explicitly augment the domain of an open ontology to include $\perp$, but this may be assumed.

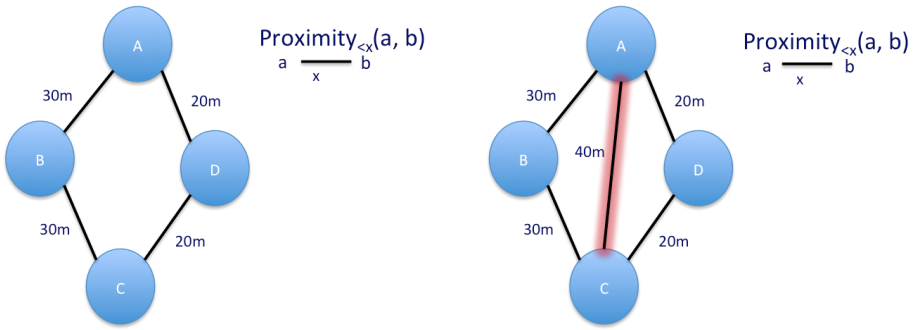

Figure 2.4: An example of logical inference for a relational proximity map.

information than $\mathbb{A}$. Applying a logical inference procedure to $\mathbb{A}$, and filling in an unknown function, relation, or constant with a concrete value can be interpreted as finding an $\mathbb{A}'$ with $\mathbb{A} \sqsubseteq \mathbb{A}'$.

It may not be possible to definitively determine whether or not an open ontology models a formula which depends on unknown functions, relations, and constants. If the true/false value of a formula depends on evaluating a function where it is unknown, an unknown constant, or the negation of a relation which is not explicitly given in the model, the formula may not be evaluated with respect to the open model. The advantage of an open ontology is the ability to evaluate formula regarding *known* information without being forced to make questionable assumptions on the unknown parts of the model.

The next section provides some examples of valid logical inference procedures for open relational ontologies. We revisit this definition in section 5.2, when we discuss inference on open ontologies performed by establishing matchings across ontologies.

Logical Inference on Ontologies

Consider the relational ontology on the left hand side of Figure 2.4. Nodes in the map represent objects or places in the world, and dark edges signify a known upper bound on the distance between them in some metric given by the weight of the edge. Since this is an open map, the absence of a black edge does not signify the converse of proximity (which we might call "anti-proximity"); if we want to express anti-proximity in this graph we must explicitly designate it with a dashed line edge.

This being a metric space, we can apply the triangle inequality to the graph and note that if A and B are within 30 meters and if B and C are within 30 meters, then A and C must be within 60 meters. Before we add this edge to the graph as shown in the right hand side of Figure 2.4, we may note that the triangle inequality applied to the edge from A to D and from D to C gives a tighter bound and express that A and C must in fact be within 40 meters of each other.

Next consider the example in in figure 2.5 with an anti-proximity edge drawn from A to

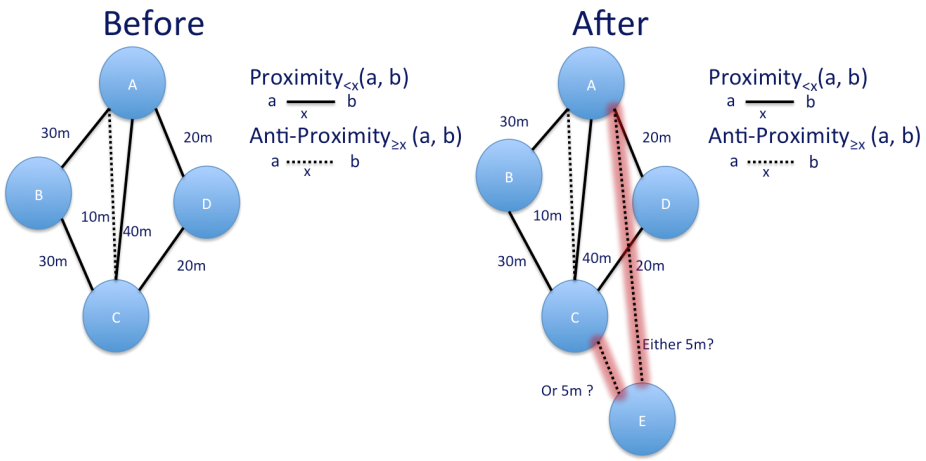

Figure 2.5: Another example of logical inference for anti-proximity

C. This indicates that A and C are known to be *at least* 10m apart, whereas the proximity edge indicates that they are *at most* 40m apart. Applying the contrapositive of the triangle inequality gives a relational ontology in which at least one of A or C must be more than 5 meters away from another node E. This matches the intuitive notion that for two objects known to be far away from each other; at least one of them must be somewhat distant from any third object. Note that this data structure is more than a simple graph now, since there is appended a disjunction between the two edges to E.

A relational ontology of distance for a Euclidean space permits more sophisticated inference methods. A considerable amount of research has been undertaken in the sensor network community to find a Euclidean space embedding for a weighted undirected graph such that the Euclidean distances between nodes in the embedding match the edge weights in the graph. If such an embedding is successfully found, it is possible to infer internode distances not explicitly specified. One such algorithm [69] uses a process of iterative trilateration with robust quadrilaterals where three nodes with known Euclidean position (say A, B, and C) are used to establish the position of a connected node (D). Once the position of D is established, it can be used in the next iteration of the algorithm as a reference point to give the position of some other node E. Other algorithms for this task are described in chapter 6.

In addition to determining unknown inter-node distances, the properties of a Euclidean space also facilitate detection of inconsistent edges signifying outlier measurements. In chapter 6 we expand upon an algorithm given in [104] which uses graph rigidity theory to identify components of a graph that admit only a specific embedding. If a questionable edge is wildly inaccurate, it can be identified by considering other rigid subgraphs that are consistent with Euclidean geometry.

These sorts of Qualitative Spatial Reasoning (QSR) received significant research attention in the 1990s. The main focus of this work was the construction of formal algebras for inference on qualitative spatial relationships. For example, Frank's calculus for cardinal directions and

informal distances such as "near" and "far" can infer such relationships for unknown cities given knowledge on how they are related to a known city network [28]. Arguably, the most notable outcome of QSR today is the Region Connection Calculus (RCC) for 2-dimensional mereology (the part-whole relationship) and topology [18]. RCC laid the foundation for the GeoSPARQL standard [9], which is today widely (but incompletely) implemented by modern semantic repositories[6] to leverage RCC relationships for queries on geospatial data sets.

Related Formal Structures from AI and Robotics

Pereira's BigActor model [74] gives a formalism for mapping with many similarities to our approach. Specifically he defines two kinds of spatial structures: a logical-space model with a rough correspondence to what we would call a relational ontology, and a physical-space model corresponding to a coordinate-based physical map. Pereira requires the same relations hold true between the same objects in physical and logical space. The model-theoretic proposal for semantic localization in this chapter can be seen as a generalization of Pereira's approach to include more diverse kinds of spatial structures.

Similar ideas to relational ontologies have been around in the world of AI and robotics research for some time. However, where semantic localization is designed to integrate modling and programming for heterogenous CPSs, the focus of research in this domain is commonly inference and autonomous decision making. As an additional point of contrast, spatial modeling in robotics is usually from the perspective of a robot as it moves from place to place, but spatial modeling for localization systems is usually from the perspective of a place as people (or robots) move within.[7]

The distinction between absolute and relative space is raised by Vieu [95]. The elements of a spatial ontology are Basic Entities (is the space composed of points or basic regions?), Primitive Notions (topology: relating to contact and part-whole relationships; orientiation: absolute, intrinsic, and contextual; distance: metric functions and discrete distance notions), and bounded/unboundedness. Vieu goes on to overview actual approaches researchers have used to represent space. Vieu also examines the difference between 3D space composed with time and 4D views.

Kuipers [47], in a classic robotics paper, introduces an ontology for spatial information flow from sensor values to, ultimately, 2-D geometry. His ontology allows information to be incomplete at different levels. For example the graph-topological connections between different maps may be known even if each of the maps hasn't been entirely fleshed out.

An example of the advantages of combining physical maps with relational information for robotic localization was demonstrated by Atanasov et al. [6]. The authors use set-based identification of semantically interesting indoor objects such as chairs and doorways

[6]Essentially a semantic repository is a database for relational data, and is discussed significantly in Chapter 4

[7]Prof. Edward Lee (my research advisor) refers to models of things moving through space as "Lagrangian Models" and models of space with things moving within as "Eulerian Models". The terminology comes from the analysis of fluid flows.

to localize their robot, instead of the more commonly used techniques that use edges and corners in the field of view without consideration for their semantics.

Conclusion

In this chapter we observed that spatial models used in cyber-physical applications frequently have good reason to be domain specific. We propose semantic localization as a unifying interface between spatial modeling and spatial programming. This abstract approach is motivated by the need to reconcile diverse spatial representations for cross-domain interaction. By treating spatial models as mathematical structures from model theory, the language of mathematical logic becomes an effective tool for describing the qualitative spatial relationships important for developing contextually aware IoT services.

This focus on semantic localization directed our discussion of physical and relational ontologies in which information may be expressed through mathematical coordinates, spatial relationships, and non-Euclidian maps of an environment. We formalized the notion of an open ontology with partially unknown information, and gave examples of logical inference on open ontologies. Open relational ontologies will be important for developing contextually aware IoT services for connected vehicles in chapter 4. We will also revisit the topic of semantic localization in the context of cross ontology inference for dynamic IoT services in chapter 5.

Chapter 3

Accessors: An Actor IoT Composition Platform

As argued in Chapter 1, the greatest long-term value of the IoT lies in emergent properties of composition and contextual awareness. Unfortunately today, the task of building a hybrid IoT service from disparate swarm systems and the task of obtaining contextual information from sensors is complicated by the heterogeneity of IoT communication. Unlike conventional internet devices, swarm devices may be deployed with application specific limitations. For example a swarm device may have to make do with low power networking technologies (like Bluetooth Low Energy or CoAP) or long range technologies (like LoRa or 5G). Even when IoT systems use compatible radios and protocols, their APIs almost always have subtle incompatibilities which require custom code to establish a working composition of services. Unless special care is taken to govern their interaction, the result will almost certainly have a nondeterministic execution.

In this chapter we introduce the **accessor platform**[1]: *an actor-based IoT composition framework designed to facilitate interaction across IoT communication silos*. The accessor platform is intended to be for the IoT what a web browser is for the internet. On the web, the browser is a single flexible platform compatible with all kinds of services. For example, when a user wants to make a web transaction with a bank, the user can navigate to the bank in their browser and request a web page that tells the browser what to display and how to perform the transaction over the internet. The web page acts as a *local proxy for the bank's remote service*. Like the bank's transaction web page, an accessor is a downloaded component which tells the swarmlet (an application which governs the interaction of swarm devices) how to perform a particular interaction with a remote service. Also like a browser, the same swarmlet can be compatible with many different remote services: it just downloads and uses each service's accessor. We give a proof of concept implementation of this idea with an augmented reality demo presented at the end of this chapter.

[1]The project website is https://ptolemy.berkeley.edu/accessors/index.html.

```
exports.setup = function () {
    this.input('inputLeft', {
        'type': 'number',
        'value': 0
    });
    this.input('inputRight', {
        'type': 'number',
        'value': 0
    });
    this.output('sum', {
        'type': 'number'
    });
};

exports.fire = function () {
    this.send('sum', this.get('inputLeft') +
    this.get('inputRight'));
};
```

Figure 3.1: Partial JavaScript code listing for TestAdder.js

3.1 The Accessors Platform

An **accessor**[2] *is a downloadable chunk of JavaScript implementing a local proxy for a remote service*, first proposed in [50]. An accessor encapsulates the complexities of communication with that service (i.e. protocol, network, API, etc.), exposing to the programmer a uniform actor interface for sensors, actuators, local machine resources, or remote web resources. In the typical pattern of accessor usage, a programmer initiates interaction with the resource represented by an accessor by providing an input to the local accessor's input port and receives a response back from the resource on the accessor's output port. Figure 3.1 gives an example of an accessor definition for an adder component. TestAdder produces the sum of the values on its input ports as output. This very simple example does little more than define the actor interface depicted in figure 3.2. A listing for a more complex accessor which uses networking capabilities is given later in figure 3.5. Interaction between remote systems can be locally coordinated by linking the input of an accessor to the output of another. Such a network of accessors is called a swarmlet, an example of which can be seen in figure 3.8.

Many of today's consumer IoT systems are **stovepipe designs**: the *components are fundamentally capable of flexible interaction with other systems, but are instead configured*

[2]The accessor pattern is commonly known as an object oriented design pattern wherein functions called getters and setters are used to change the properties of an object, but this is unrelated to our use of the term "accessor" in this thesis.

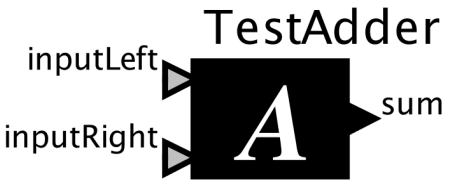

Figure 3.2: The actor interface for the TestAdder accessor

for a specific non-interactive service. A typical network architecture of a Stovepipe Internet of Things (SIoT) system is depicted[3] in Figure 3.3 with circles representing Swarm/Fog/Cloud nodes and lines representing communication channels. SIoT Swarm devices are connected to a fog gateway which provides internet connectivity to a cloud back end specific to the SIoT service. To control the SIoT's swarm devices, a human typically downloads a mobile app and sends instructions up to the cloud which are then routed back down through the gateway to swarm devices. Interaction may be possible between SIoTs at the level of services, which is accomplished through establishing communication through cloud back ends. This can result in the rather strange situation where devices on the same network which are literally inches apart from each other, are forced to coordinate their interaction in data centers hundreds of miles away.

Accessors enable an alternate network architecture depicted in Figure 3.4. In an Accessor Internet of Things (AIoT) architecture, swarm devices are viewed as reusable and repurposable resources. Instead of being tied to a particular gateway and cloud backend for a particular service, swarm devices are coordinated to accomplish some task by a swarmlet running in the fog. That swarmlet may utilize cloud capabilities for storage and big data analysis, but it does not necessarily depend on cloud availability to work correctly. We introduce a design for a fog-based user interface app for swarmlet management in chapter 4, and present a design methodology for the direct composition of swarm devices into automatic services without a human in the loop in chapter 5.

The AIoT architecture has advantages for security. The Secure Swarm Toolkit[4] developed by Kim and Lee [45] leverages the AIoT architecture for authorization in IoT systems. Systems for authorization, the task of authenticating a user's identity and verifying their permission level for use of a particular resource, are frequently implemented as an SIoT architecture with a single trusted server in the cloud. If that cloud server is taken over by

[3]The specific SIoT and AIoT network topologies depicted in figures 3.3 and 3.4 are illustrative, and meant to highlight the differences between the vertically integrated approach of the SIoT and the swarmlet focused design of the AIoT. It is for instance not a requirement that an AIoT system must interact with exactly two swarm devices. The point is an AIoT system is capable of interacting with independtly manufactured swarm devices.

[4]Available at github.com/iotauth

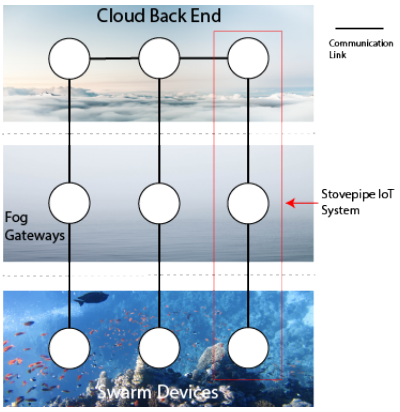

Figure 3.3: Stovepipe IoT Network Topology.

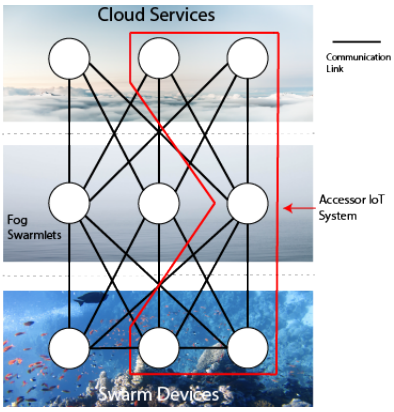

Figure 3.4: Accessor IoT Network Topology.

a malicious adversary or if a network connection is not available from the fog to the cloud, the entire authorization system will likely fail.

Instead, Kim and Lee propose a *locally centralized, globally distributed* authorization infrastructure leveraging the AIoT architecture by moving authorization services to the fog. Their approach uses a local authorization entity (Auth) to provide authorization services for a particular fog computing environment, while a global network of Auths manages trust relationships across fog environments. The Secure Swarm Toolkit is integrated with the accessor platform and avoids the pitfalls of a single point of failure.

Accessor Specification

As local proxies for remote services, Accessors are meant to be downloaded from the web (or directly from the proxied remote service) and locally evaluated. The accessor specification[5] version 1.0 uses JavaScript as the implementation language to take advantage of JavaScript's execution within a controlled host environment. It is somewhat safer to download and evaluate JavaScript than some other language because a JavaScript program only has access to the specific host capabilities implemented by its environment. For example if functions for file system I/O are not provided by the host, I/O will be unavailable to any JavaScript program executing in that environment. Usually JavaScript on the web is downloaded and run in a browser where it interacts with DOM elements, but accessors run on an accessor host (see section 3.2) where some accessor-specific functions must be defined and others are made available for use.

If an accessor defines certain functions (`exports.setup`, `exports.initialize`, `exports.fire`, and `exports.wrapup`), it can expect the host to call those functions at certain points in its life cycle. The function `exports.setup` will be called first when the accessor is instantiated to set up its actor interface with inputs, outputs and parameters.

[5]See https://wiki.eecs.berkeley.edu/accessors/Version1/AccessorSpecification for the full document.

The `exports.initialize` function is called when the swarmlet begins running and can be used to set up computational resources such as network connections. `exports.fire` is called by the host when an input arrives at the accessor, or whenever the host wants the accessor to respond to its inputs. As an alternative to `exports.fire`, an accessor may use its implementation of `exports.initialize` to set up input handler callback functions tied to particular input ports. Finally, `exports.wrapup` is called by the host when the swarmlet is shutting down so the accessor can clean up and release the computational resources it acquired in `exports.initialize`.

All accessors must at minimum implement an interface of inputs, outputs and parameters. All three may be defined with a type (boolean, int, number, string, or JSON). When the accessor host provides a value to an input port, it may be consumed by the accessor via the `get('InputName')`. Calling `get('InputName')` when no input is available, results in a null value or a specially designated default value if a default has been declared. Parameters are similar to inputs, but with persistently available values available via the `getParameter('ParameterName')` function. Unlike an input, a parameter is meant to be somewhat static over the lifetime of the accessor once set via `setParameter('ParameterName', ParameterValue)`. Both inputs and parameters may be defined with default values. An accessor may send data to an output port via `send('OutputName', OutputValue)`. Outputs may be labeled as spontaneous or otherwise to support analysis for the detection of causality loops within directed cycles of connected accessors.

Accessors may be assigned interface, subclass, and container relationships with other accessors. The SoundActuator in Figure 3.5 does not explicitly define inputs, outputs, and parameters because it implements the ControllableSensor interface. When `implements('InterfaceName')` is invoked in an accessor's `exports.setup` function, that accessor is given the intputs, outputs, and parameters defined by InterfaceName. In the case of SoundActuator, the ControllableSensor interface gives it the "control" input and the "data" and "schema" outputs seen in Figure 3.6. Similar to an interface, a subclass accessor specified with `extend('BaseAccessorName')` inherits the inputs, outputs, and parameters of BaseAccessorName, but additionally a subclass accessor also inherits all exported fields and functions from BaseAccessorName. A **composite accessor** (like SoundActuator) is *an accessor containing an internal network of accessors.* A subaccessor is created via `instantiate` and its input and output ports may be connected into an internal accessor network via `connect`.

The core accessor interface is intentionally minimalist and does not include networking or hardware capabilities. Minimalism for core functionality makes it easier for accessor hosts (see section 3.2) to be implemented on platforms with restricted capabilities. However, additional capabilities may *optionally* be provided by a host in modules such as "cameras" or "http-client". An accessor may use a host module by importing it with `require('ModuleName')` and calling functions from the module's API on the returned object. Such a call to `require` in an accessor will fail if the host does not implement ModuleName, but will succeed on any host which correctly supports the module's API. This

```
exports.setup = function() {
        this.implement('ControllableSensor');
        var WebSocketClient = this.instantiate('WebSocketClient',
            'net/WebSocketClient');
        WebSocketClient.input('server', {
            'value': '128.32.47.81'
        });
        WebSocketClient.input('port', {
            'value': '8078'
        });
    this.connect('control', WebSocketClient, 'toSend');
    this.connect(WebSocketClient, 'received', 'data');
};

exports.initialize = function() {
        // At initialize, send the schema;
        this.send('schema', schema);
        // Also send null data.
        this.send('data', null);
};

// NOTE: Using "options" instead of "choices" below will result in
// a pull-down list rather than a radio button.
var schema = {
  "type": "object",
  "properties": {
    "sound": {
        "type": "string",
        "title": "Name of the sound to produce",
        "description": "The name of the sound to produce",
        "choices": ["strum", "bell",
            "laughter", "thunder", "train", "ring"]
    }
  }
};
```

Figure 3.5: Partial JavaScript code listing for SoundActuator.js

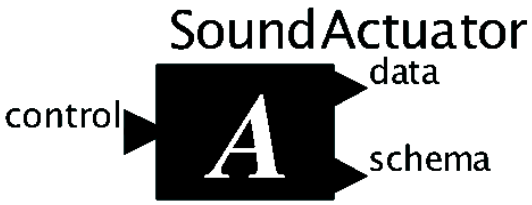

Figure 3.6: The actor interface for a ControllableSensor accessor like SoundActuator.

design keeps accessors decoupled from module implementation and portable across all hosts which support the modules they need.

At the time of this writing, a new effort [66, 65] within the accessor development group is underway to create a generalization of accessors called "reactors". A reactor defines an actor interface and reaction rules for inputs in a new polyglot language called *Lingua Franca*. Reaction rules in Lingua Franca are written in a separate Turing complete language. For example, while a reactor written with JavaScript reaction rules may be compiled to an accessor, C language reaction rules will be transformed into a new experimental C accessor specification.

Swarmlets

A network of accessors, called a swarmlet (eg. Figure 3.8), may be used to coordinate the interaction of swarm devices and cloud services. As accessors represent local proxies for remote services, connecting accessor inputs and outputs in a swarmlet has the effect of relaying information between the proxied services. However, giving semantics for the execution of accessors in a swarmlet is a surprisingly subtle endeavour. One of the greatest strengths of the accessor platform is its ability to leverage a programming model combining Asynchronous Atomic Callbacks (AAC) with a deterministic Actor-based Discrete Event (DE) Model of Computation (MoC) and Deterministic Temporal Semantics (DTS). Determinism is an important property of a model because it ensures a single well-defined correct behavior, facilitating verification and testing. In this section we will present the concepts behind the aforementioned list of acronyms, and explain how they can be used to give swarmlets well defined temporal and concurrent behaviors.

But first it is important to identify the types of communication occurring in a swarmlet. Accessors with connected input and output ports communicate with each other over a **horizontal contract** which *governs their interaction at the same level of abstraction* [64]. An accessor also communicates with its proxied service over a **vertical contract** *governing its interaction across levels of abstraction.* In other words, the specific networking technologies used to implement an accessor constitute its vertical contract. When accessors are wired together as black boxes, those connections constitute a horizontal contract. The differences between vertical and horizontal contracts in a Swarmlet are depicted in Figure 3.7.

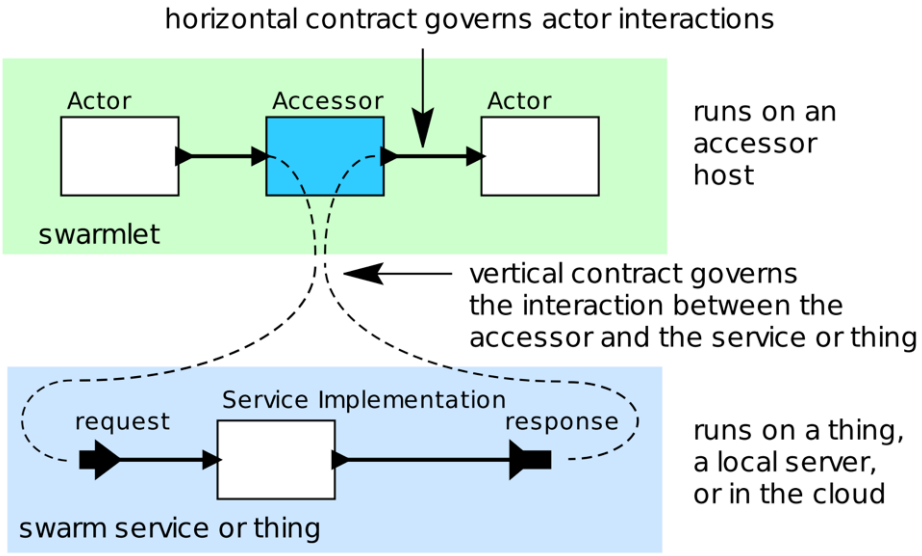

Figure 3.7: An illustration of vertical and horizontal contracts in the accessor pattern. The original source of this image is Figure 1 of [50], with different annotations courtesy of a 2017 presentation by Victor Nouvellet.

.

JavaScript environments like Nashorn, browsers, and Node.js execute JavaScript in a a single threaded event loop. Taking Node.js as an example, AACs are added to a queue of callbacks as incoming (vertical) connections or timer events occur. The callbacks are asynchronous because they are first registered with the environment and later triggered in response to an event. Due to JavaScript's single thread of execution, a callback has an atomic execution which will not be interrupted by another callback. AAC's are useful as a comprehensible way to program concurrent event-driven applications, but come with the drawback of not allowing potentially valid interleavings of callbacks on parallel hardware.

Internally, an accessor's vertical AACs are triggered by network events. Externally an accessor presents a horizontal actor interface and may be triggered by values arriving on input ports. Actors have exclusively internal (i.e. not shared) state and only communicate with other actors along communication channels linking input and output ports. Actors were first proposed by Hewitt in 1973 [37] as a concurrent MoC and since then have been developed into a variety of formalisms and languages (eg. Agha [2] or Akka [34] actors). Accessors are based upon the Ptolemy II flavor of actor with formal semantics defined in [93] possessing an executable actor interface: the functions fire, postfire, deadline, and time-update. A network of Ptolemy II actors (or a swarmlet, if those actors are accessors) is coordinated by a director which implements a concurrent MoC for the actor network.

The Ptolemy II (and accessor platform) variant of DE is a timed MoC which establishes semantics for the temporal relationships of (horizontal) messages between accessors [79]. Timestamped messages are central to the timed MoC concept. Erroneous behavior can

result in a swarmlet which processes timestamps out of order or incorrectly allows execution to proceed while ignoring logically simultaneous messages. The DE MoC ensures these problems do not occur and that execution is deterministic by respecting timestamp order precedence relations in the swarmlet topology. A DE director for a swarmlet will only cause an accessor X to react to its inputs when there are no unprocessed messages with earlier timestamps, and all upstream accessors which may produce an earlier message for X have reacted first. The execution of a swarmlet may be synchronized to real time by adding the additional requirement that wall clock time must have caught up to a message's timestamp before it may be delivered. This requirement ensures a delay actor which adjusts the timestamp of a message will cause a delay in real time.

Embedding AACs in concurrent accessors of a DE swarmlet brings the best of both worlds. Within an accessor, single threaded event driven programming governs execution of its vertical contracts, while the naturally expressed concurrency of a DE swarmlet governs parallelizable horizontal contracts. Lohstroh and Lee formalized the composition of AAC with DE using interface automata [64]. The asynchrony of JavaScript is nicely contained within the internal state of an accessor and the overall behavior of the network is clearly defined and deterministic with respect to horizontal contracts. But what about the temporal semantics of vertical contracts?

On its own the JavaScript event loop does not provide any mechanisms to ensure the correct temporal ordering of callbacks. Setting a JavaScript timeout only ensures a callback will be triggered at some point *after* the designated interval. For two timeouts with durations N and P, even if $N < P$ there is no guarantee the timeouts will occur in the expected order. As out-of-order control can be disastrous for the embedded system vertically proxied by an accessor[6], Jerad and Lee introduced a mechanism to provide DTS for accessors [43]. Their approach introduced an accessor API for labeled logical clock domains which schedules timeouts for deterministic callback execution. Taken together AAC, DE, and DTS make swarmlets a precise tool for coordinating the behavior of swarm devices.

3.2 Accessor Hosts

As discussed in 3.1, an accessor must define and use functions provided by their host environment. An **Accessor Host** *provides a JavaScript runtime environment with implementations of core (and optional) functions from the accessor specification and manages the life cycle of a swarmlet.* In other words, an Accessor Host is the program which runs a swarmlet. So far we have implemented five distinct hosts with different capabilities.

The core functionality shared between all hosts is factored into a single JavaScript file named CommonHost. To maintain compatibility with all hosts, the Common Host is writ-

[6]As an example of how bad out-of-order control can be, consider a robot driving toward a cliff with intermittent network access to its control station. The first control message says, "Keep driving, everything is fine." The second control message says, "Stop!" If those messages arrive out of order, the robot will briefly stop and then commence driving over the edge.

ten in pure JavaScript and implements universal features like instantiating and connecting accessors within a composite. At this time CommonHost consists of approximately 3200 lines of heavily commented code and could be minified to be much smaller.

The hosts listed below are responsible for setting up a JavaScript runtime, loading CommonHost, and providing bindings for core CommonHost functions appropriate for the platform such as `require` and `getResource` for file system access. A host may also implement optional modules to provide accessors with host capabilities like cameras or networking. A module has a uniform API across all hosts, but each host is responsible for the native module implementation which runs on its platform. This design allows accessors to be portable across hosts.

CapeCode

The CapeCode host is a Java-based offshoot of the open source Ptolemy project[7] which was originally developed as a testbed for MoC research on Actor-based modeling of CPSs. A Ptolemy II model consists of an executable network of actors and a director which governs the execution by implementing a MoC. Notably, Ptolemy II provides a GUI and graphical programming language, making CapeCode a strong development environment for Swarmlets.

A CapeCode swarmlet is implemented by embedding accessors within Ptolemy II actors. Ptolemy II provides a special JavaScript actor which uses the Nashorn JavaScript engine to provide a runtime for the contained code. Accessors can be combined into a CapeCode swarmlet by wiring together their actor ports under a Discrete Event director. CapeCode provides a code generation capability which converts a swarmlet written this way into a composite accessor compatible with other hosts.

CapeCode native modules are written in Java, and several implementations are based upon Ptolemy II networking actors. In addition to its capabilities as an accessor host, Ptolemy II has an extensive library of Java actors including the PILOT machine learning toolkit [3]. PILOT provides actor-based implementations of Dynamic Bayesian Network learning algorithms which may seamlessly be combined with a CapeCode swarmlet[8].

Node

The Node host is based upon the popular Node.js[9] JavaScript run-time environment for server-side scripting. Node.js uses Chrome's V8 JavaScript engine to build high performance network applications. Node.js is compatible with the powerful Node.js Package Manager (npm) which provides many native implementations of host capabilities for Node host modules. The Node host is written as a Node.js module, which may be loaded into a Node.js program to instantiate the top level composite accessor of a swarmlet. Due to the Node

[7] https://ptolemy.berkeley.edu/

[8] The resulting actor network will not however be a swarmlet because it contains PILOT's Java actors.

[9] nodejs.org

hosts's proximity to the active npm development community and high performance platform, we view it among other hosts as the best choice production environment for swarmlets.

Browser

Where CapeCode is a development environment and Node is a production environment, the Browser host is a good choice for creating accessible demonstrations of swarmlets. The Browser host consists of a web page with the JavaScript for the host embedded in a script tag. The host has a built in capability for loading a library of accessors and creating a web form user interface for the inputs, outputs, and parameters of an accessor. We use the Browser host for the accessor website[10] as it is the most accessible way for a user to immediately start using accessors.

The Browser host is restricted by the capabilities of a browser and consequently has limited file system access and support for optional modules.

Cordova

The Cordova host is a more experimental host built upon Apache Cordova[11], an open source project for building apps which can be deployed with the same code across browser, Android, and iOS platforms. Like the Browser host, the Cordova host is embedded within the script tag of a web page, but the page may then be converted into a native app via the Apache Cordova toolchain. Key to this process is the use of a Cordova-enabled WebView, the programming class used to implement a mobile browser. Although the accessors of a Cordova swarmlet run within the WebView, the swarmlet is built inside a fully fledged app *containing* the WebView. Consequently, the app has access to native capabilities outside the restrictions of a browser environment. Apache Cordova provides these capabilities (such as accelerometers, location services, and device storage) through a library of plugins and an active community of developers have created additional plugins with limited cross-platform support. Therefore the Cordova host has access to a unique set of mobile-specific modules with native implementations provided by plugins.

Unlike the Browser host, the Cordova host has a minimalist user interface consisting of a status display and a text display for accessors to print debugging information. In chapter 4, we give an architecture for developing more sophisticated app-specific user interfaces and discuss additional challenges in building a Cordova swarmlet in section 4.4.

The Cordova host is subject to some unique limitations, justifying its experimental status. A significant number of low-level plugins are not truly multi-platform: a plugin may only work with Android or iOS but not the other. Relatedly, some phone UI capabilities like camera preview can't be rendered at decent frame rates in the WebView. In this situation a Cordova swarmlet may need to launch a separate mobile activity for the special UI element, breaking multi-platform compatibility.

[10] ptolemy.berkeley.edu/accessors/library

[11] cordova.apache.org

The mobile application lifecycle imposes other unique limitations on a Cordova swarmlet. For example, swarmlets on hosts with non-mobile operating systems can expect to run in the background without interruption, but this is not the case for a mobile application swarmlet. A mobile operating system may move a swarmlet out of memory when it is minimized to free up space for other apps. As of the time of this writing, the Cordova host wraps up a swarmlet when it's minimized and restarts it (from the beginning) when the user navigates back to it. However it may be desirable in the future to extend the Cordova host with support for persistent background processes and the ability to save the state of a swarmlet when minimized so it may later be resumed at the same point of execution.

Mobile operating systems restrict system capabilities through a permission system, meaning swarmlets using certain modules must specifically request access to capabilities like location services and have those requests granted by the user before they can be made available. For example, an Android swarmlet with dynamic capabilities must declare all the permissions it may potentially need to use in its AndroidManifest.xml file even if it may never actually use those permissions.

Duktape

The Duktape host is an experimental host which serves as a proof of concept for running swarmlets on resource constrained embedded hardware. The Duktape engine[12] provides a C/C++ JavaScript runtime environment. We succeeded in using the Duktape host to run a simple program to display integers on a Maxim Integrated MAX32630, which is a Cortex-M4 with 512K RAM and 2Mb flash [15]. The resulting executable was (in bytes)

- text: 291 848—program code in flash
- data: 2964—initialized data in RAM
- bss: 8400—uninitialized data in RAM

At the time of this writing there are no significant Duktape host modules, but they would be implemented in C or C++.

3.3 Component-based Augmented Reality Design

Swarm interactivity in the IoT currently faces a chicken and egg problem: The lack of compelling commercial applications for interactive systems is a result of weakness in infrastructure, and the infrastructure for contextual awareness has not been developed because apps have so far been contextually uninteresting. Augmented Reality (AR) has the potential to be a unifying application for the IoT with the opportunity to become the egg which begins a virtuous cycle of IoT development. In this section we describe a component-based

[12] duktape.org

architecture shown in Figure 3.8 for an AR swarmlet with the dynamic capacity to connect to newly discovered devices.

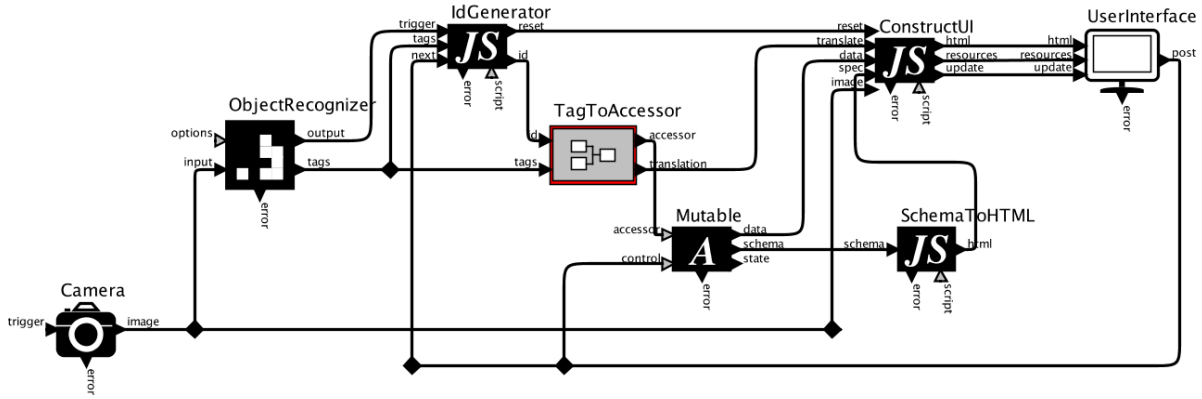

Figure 3.8: The augmented reality controller swarmlet.

The application itself is depicted in Figure 3.9 where the control interface for an IoT device (in this case a Turtlebot3 Waffle robot) is superimposed directly over a live video of the device. Devices are identified by April Tags [72] which are essentially large QR codes for object detection in robotics and computer vision applications. An April Tag encodes a number which is used as the identifier for a specific device. If multiple tags are present in a frame of video, the next tag can be selected for control by pressing the "Next Tag" button at the top of the image. Commands can be sent to the device through its accessor by selecting an option on the "Inputs:" form and pressing submit. Data from the device's accessor is reported at the top of the form. Figure 3.9 shows a robot with movement commands and wheel velocity data, but we have also implemented a similar interface for a Philips Hue and LIFX smart light bulbs, Bluetooth Low Energy Environmental Sensors (BLEES) [1], a connected speaker capable of playing audio clips (accessed through the SoundActuator of Figures 3.5 and 3.6, and a text-chat display.

Augmented Reality Swarmlet

The swarmlet controller for the AR system is depicted in Figure 3.8 and runs on the CapeCode host. In this section we will elaborate on each of the accessors in the controller.

The Camera accessor uses the cameras module to interface with a camera device on the host platform. The cameras module automatically detects available webcams or usb cameras on the host, and creates a list of available devices for selection in a Camera accessor parameter. Camera may be triggered to produce an individual frame of an image on its output port, or automatically produce images at a specified rate. These images are sent directly to the ConstructUI where they are used as the background of the user interface and also to ObjectRecognizer.

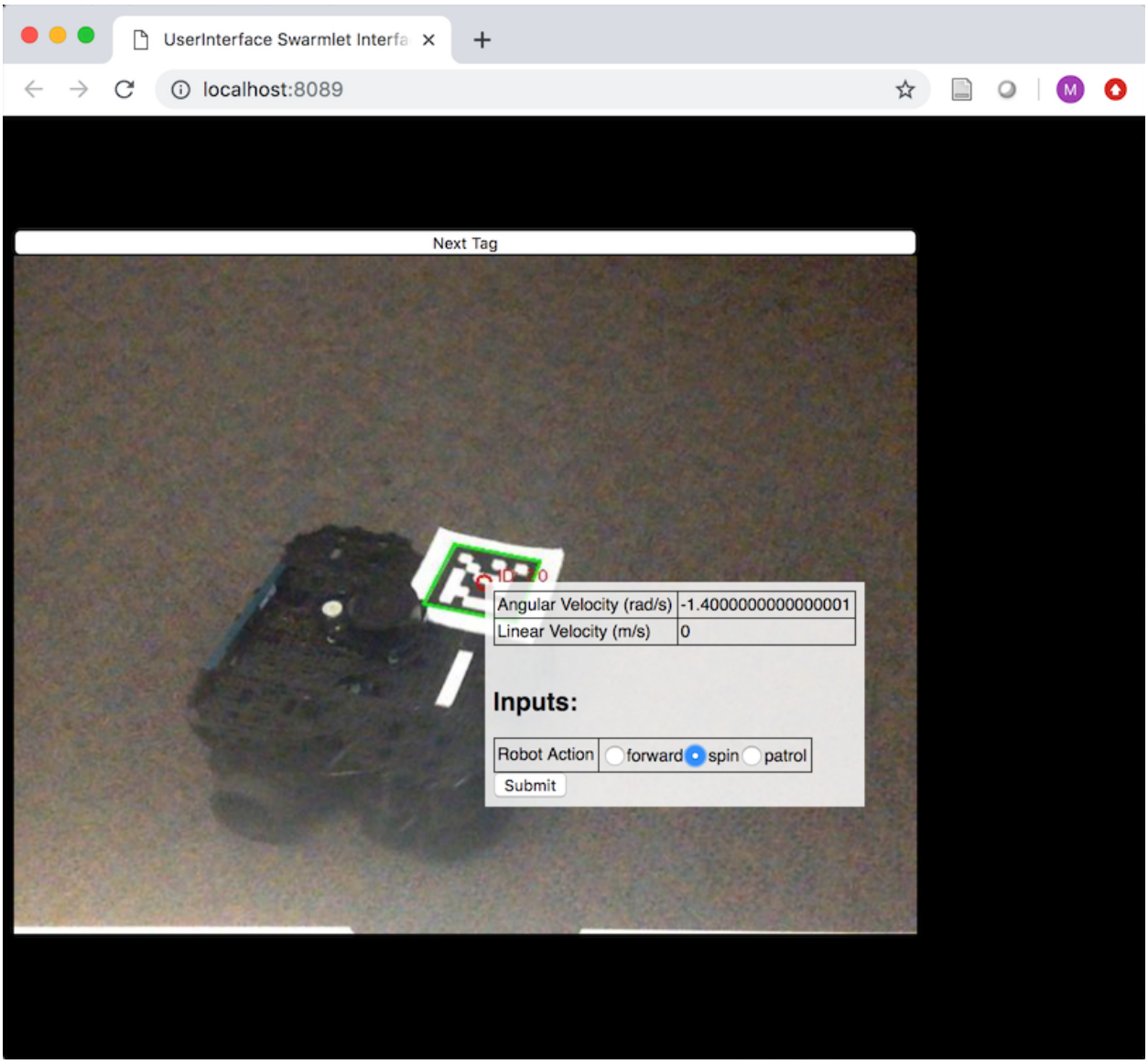

Figure 3.9: Screenshot of augmented reality user interface. The robot (a Turtlebot3 Waffle) has been issued a "spin" command and reports its changing angular velocity.

ObjectRecognizer uses an April Tags module to identify the location and encoded ID number of each visible tag in the image. It produces an array of tag information on its output. IdGenerator determines which tag (if any) in the array of visible tags the user has selected to focus upon for control. It accomplishes this by sorting visible tag IDs numerically, and iterating to the next least tag when UserInterface reports the user has pressed the "Next Tag" button. If IdGenerator has received an empty list of tags for a given number of frames, it issues a reset command so ConstructUI knows to clear the control form for whatever device was last displayed.

TagToAccessor is a composite accessor responsible for obtaining the accessor for the selected device. If the selected ID from IdGenerator is visible in the image, TagToAccessor extracts the location in the image of the corresponding tag and sends it to ConstructUI. If the ID was different from the last ID TagToAccessor received as input, it downloads the corresponding device accessor from a key-value store service. It is the responsibility of a device to register itself with the correct (ID, accessor) pair in the key-value store when it becomes available to the AR system (step 1 of figure 3.10). TagToAccessor produces the downloaded accessor code for the Mutable accessor. If the previous tag has been removed from the image and there is no replacement, TagToAccessor produces an empty message instead of an accessor, indicating the Mutable should be reset.

A more advanced TagToAccessor implementation would use a location-based discovery service to dynamically obtain the tag to accessor matching for the user's current environment and update the matching when new AR tags are deployed. The accessor itself could come from a web service or from a local edge computer that is aware of devices in the local environment. These possibilities are explored at length in chapter 4.

The mutable accessor is probably the most interesting accessor in the AR swarmlet. It is a special higher order accessor which takes the code of an accessor as input, and instantiates it. The newly reified (i.e. made real) accessor is connected to the mutable's input and output ports of the same name, as depicted in figure 4.7. For this to work, the mutable's interface must satisfy an abstraction relationship with the reified accessor. This relationship is guaranteed in the AR swarmlet by defining the ControllableSensor accessor interface from Figure 3.5 and 3.6 with "control" input and "data" and "schema" output, and implementing ControllableSensor in the `exports.setup` function of the mutable. With respect to ControllableSensor inputs and outputs, the mutable accessor is essentially replaced by the reified accessor.

A ControllableSensor accessor has the behavioral expectation of producing a self-described user interface for itself on its schema output upon initialization in the mutable. The schema is converted to an HTML web form by the SchemaToHTML accessor and superimposed over the camera image at the location of the corresponding April Tag in ConstructUI.[13] ConstructUI combines its various inputs together into HTML and image resources for UserInterface.

The UserInterface accessor launches a web server and automatically opens a browser to the hosted page. When UserInterface receives an update from the swarmlet (to for example display the latest image frame), it modifies the host page via a web socket connection. When the user interacts with the form for the selected Controllable Sensor on the page by clicking "submit", the resulting POST request is directed back to the control input of the Mutable. The ControllableSensor reified within interprets the command as instructions to perform some action on the device such as moving the robot or playing a sound. Feedback from the device, for example robot velocity or sound playing status, is produced on the

[13]This notion of self-described accessor interfaces is expanded from HTML form elements to more general web components in chapter 4.

ControllableSensor's data output and passed on to ConstructUI where it is rendered to the AR display.

The network architecture of this augmented reality system and the steps involved in controlling a device from the AR interface are diagrammed in Figure 3.10. As illustrated, a new device must first register its accessor and its AprilTag ID with the Key-Value Store (1) before it can be identified by the AR Swarmlet (2, 3, and 4) or controlled by its accessor (5 and 6).

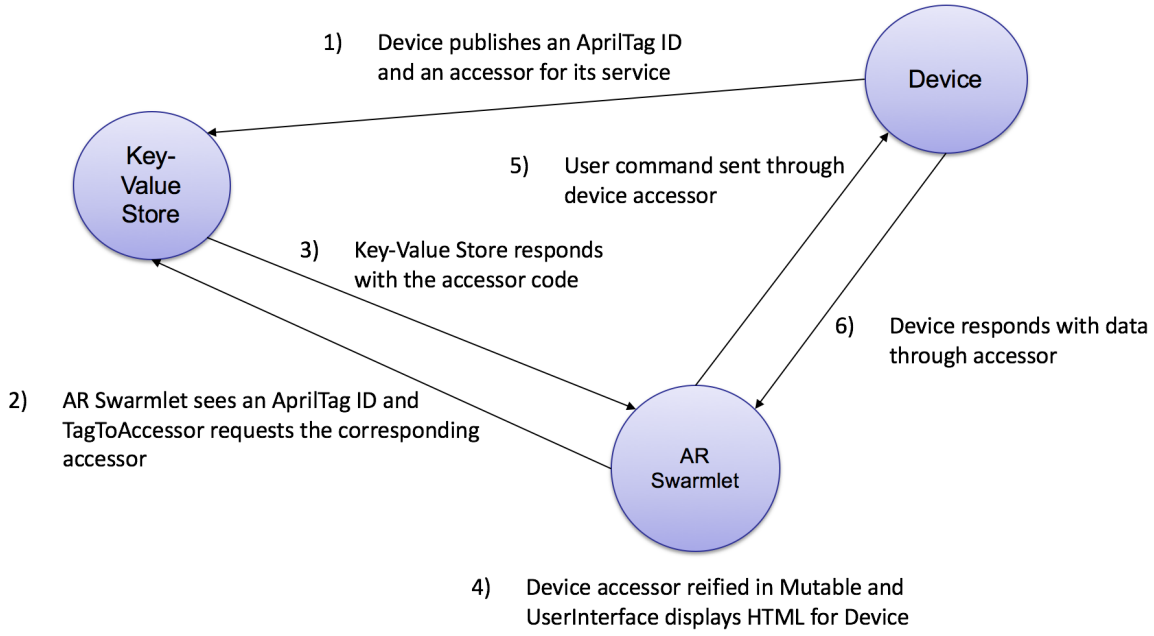

Figure 3.10: Network architecture of augmented reality system.

3.4 Related IoT Composition Platforms

Due to the clear need for composition in the future of the IoT, many platforms besides accessors have been developed for this purpose. In this section we survey and compare them to the accessor platform.

Voice assistants have emerged as an immensely popular IoT device for linking home users with cloud services. The companies behind the most popular voice assistants have developed smart home IoT composition platforms centered upon voice assistants such as Google Home,[14] Amazon Alexa's Smart Home skill,[15] and Apple's HomeKit for Siri.[16] While

[14]developers.google.com/actions/smarthome
[15]developer.amazon.com/docs/smarthome/understand-the-smart-home-skill-api.html
[16]developer.apple.com/homekit

voice assistants offer a unique interface for IoT interaction, the underlying infrastructure behind their composition platforms are built according to the SIoT topology from Figure 3.3 and lack the advantages of the AIoT topology as discussed in section 3.1.

IFTTT, an acronym for If This Then That is another cloud-based service composition framework. IFTTT allows users to develop "applets" which allow programmers to set predicates on actions taken on web services such as Instagram, Evernote, Reddit, and Spotify. For example, whenever a user likes a photo on Instagram, an IFTTT recipe (that's what they call their trigger-action pairs) could automatically send an email as the action. IFTTT also supports programmed reactions to input devices based on cloud services, such as the speech recognition of the Amazon Echo, and control of devices that have web interfaces, such as smart thermostats. While IFTTT promotes a more cross-vendor service composition than the voice assistant centered technologies, the result is still an SIoT Topology where interaction happens at the cloud.

Home Assistant[17] and openHAB[18] are two open source home automation systems which are designed to run on a Raspberry Pi or a local server, bringing the advantages of an AIoT Topology. Unlike the previous cloud-centric systems, Home Assistant and openHab support a variety of communication protocols for directly interacting with swarm devices. In both platforms a developer can set up triggers for home automation capabilities, but the unspecified semantics for simultaneous triggering and inability to reason with clear temporal semantics or concurrency model results in a programming environment unsuitable for the precise embedded system control of which accessors are capable.

For both Home Assistant and openHAB, the server is the center of attention: devices define channels and communication protocols through which the server can obtain information or trigger an action. Publish-subscribe networking protocols like MQTT and ROS may be used to similar effect. Although the network architecture of a swarmlet has an analogous pattern in which the swarmlet is a hub for device control, accessors present a different actor-based programming model in which data flows directly from one local proxy to another rather than conceptually arriving at and then leaving the accessor host. Downloadable accessors provide opportunities for code reuse at the level of devices, rather than protocols and channels.

Similar actor oriented platforms for IoT composition are in development as well, notably Calvin [75] and Node Red [30]. Aside from syntactic differences, accessors bring deterministic models of computation from the timing-critical world of embedded systems [43] and interface theory [64]. For example, Node Red's "flow-based programming" (which is a variety of dataflow programming; an actor MoC) lacks a deterministic discrete event MoC ensuring the correct delivery of timestamped messages at components. Additionally, accessors have a greater focus on dynamic discovered component behavior via the mutable accessor [15] (see Section 4.4) and integration with semantic technologies discussed in chapter 4.

[17] www.home-assistant.io

[18] www.openhab.org

Conclusion

In this chapter we gave an overview of the accessors platform for IoT composition. Accessors were developed to facilitate interaction across IoT communication silos. When JavaScript files matching the accessor specification are downloaded and wired together into a swarmlet, the resulting network executes with a well-defined and deterministic MoC. Thanks to the module system for host-specific capabilities, swarmlets are portable across accessor hosts implemented on a variety of platforms.

The accessors platform is a powerful tool for developing IoT services with modular components such as the augmented reality demo. Elaborating on this idea, we explore ontology based mechanisms for discovering relevant accessors for intelligent vehicle services in chapter 4. Although accessors encapsulate the complexity of communication with a remote service, the spatial/contextual ontology for data produced or expected by an accessor may be inconsistent with the rest of a swarmlet. To address this problem of semantic interpretation we introduce accessor-based components called adapters in chapter 5.

Chapter 4

Semantic Accessors for the Connected Car

Connected cars have the potential to transform a vehicle from a transportation platform to a platform for integrating humans with a city. To that end we introduce semantic accessors (actor based local proxies for remote services) as a novel, and powerful discovery mechanism for connected vehicles that bridges the domains of Internet of Things (IoT) composition frameworks and the semantic web of things. The primary components of this approach include a local semantic repository used for maintaining the vehicle's perspective of its real-world context, accessors for querying and dynamically updating the repository to match evolving vehicular context information, accessors for services (such as parking) linked to a service ontology, and a swarmlet controller responsible for managing the above in accordance with user input. We demonstrate this semantic accessor architecture with a prototype dashboard display that downloads accessors for new services as they become available and dynamically renders their self-described user interface components. While these technologies were developed with the connected car platform in mind, they are also applicable in a variety of dynamic IoT settings.

The semantic accessor architecture presented in this chapter was built as an extension of the component based augmented reality design from chapter 3. Both designs dynamically download and reify accessors for local services in the environment, but they are different in three major respects:

1. Services are identified by different mechanisms. Augmented reality uses AprilTag detection to select a service while the dashboard selects a service by contextual ontologies stored in a semantic repository.

2. The semantic accessor architecture has the ability determine whether or not a service will be compatible. All visible services are assumed to be compatible in the augmented reality system.

3. The dynamic user interface prototyped in this chapter allows accessors to self-describe a more expressive user-interface. The augmented reality interface only supports an HTML form UI. The Dashboard supports web components, a standard for modular web page elements which allow for arbitrary bundles of HTML, CSS, and JavaScript. Web components allow for more aesthetically sophisticated and interactive user-interface elements.

4.1 Connected Cars in the Connected City

With global automotive sales on a downward trajectory, car manufacturers are seeing enormous opportunities in the areas of electrification, autonomy, vehicle ownership and a gamut of connected services [92][7]. Connected cars have the potential to transform a vehicle from a transportation platform to a platform for integrating humans with a city. Traffic lights and signage (static and dynamic) have served this role in the past, but these are impossible to personalize and difficult to make sufficiently adaptive and dynamic. Here is a story from the connected city:

> As I am approaching downtown Berkeley, I want to know more than that it is safe to drive through the next intersection (a green light), but also that the garage I intended to park at is full, and that I can still make my 2PM meeting on time if I head for an alternate garage and take a tram one stop. When the battery charge on my car cannot get me to my destination, I want alternatives, where if I'm flexible on timing I can recharge while eating lunch, or if I'm not, I can swap batteries for a higher fee. If I'm getting drowsy, I want to know where to get a good macchiato or a brisk walk around a pond.

Simple solutions like putting a smartphone into the vehicle's dashboard miss enormous opportunities. The car has context that a phone does not, and its solutions can seamlessly move from one user to another, a feature that may become increasingly important as personal ownership of vehicles declines.

We identify and propose solutions to three main challenges from the vehicle's perspective in making this vision a reality. 1) Once a driver has established the intention of seeking out the service (eg. parking), how does the connected vehicle select the most contextually appropriate service from a service library? 2) How does the vehicle communicate with this service, with which it has never before interacted, using the correct radio, protocol, and API? 3) How does the vehicle coordinate its use of the service with on-board sensors or other similarly discovered services?

We propose a semantic accessor architecture for connected car integration to address these challenges. This semantic accessor architecture is a novel combination of semantic technologies, as commonly used in the Semantic Web, with the Accessors project (chapter 3), an open source platform for remote service communication and composition. The primary components of this approach include a local semantic repository used for maintaining

the vehicle's perspective of its real-world context, accessors for querying and dynamically updating the repository to match evolving vehicular context information, accessors for services (such as parking) linked to a core service ontology, and a swarmlet controller responsible for managing the above in accordance with user input. Our prototype of this system is available on the iCyPhy repository (https://github.com/icyphy/accessors) under a BSD license.

In particular the main elements of this chapter include:

- An ontology for the accessor and vehicular service subject domains;
- New semantic accessor components which integrate semantic technologies with the accessors platform;
- A proof of concept implementation of the semantic accessor architecture; and
- A self-describing user interface paradigm for accessors leveraging web components to generate a dynamic interface.

Taken together, these contributions tell an end-to-end story through which dynamic vehicular context information is obtained and used for service discovery, the service's self-describing user interface is rendered to the vehicle's Head Up Display (HUD) or In-Vehicle Infotainment (IVI) displays (jointly referred as Dashboard), and interaction with the remote vehicular service is commenced in accordance to its arbitrary third-party API. By combining semantic technologies with accessors we have developed a novel, and powerful discovery mechanism. We begin in Section 4.2 with an overview of semantic technologies and their applications to automobiles, present our work on integrating semantic technologies with the Accessors project in Section 4.3, demonstrate the Semantic Accessor Architecture in Section 4.4, and conclude in Section 4.5.

4.2 Background and Related Work

Future connected and autonomous vehicle application scenarios span a multitude of domains including but not limited to Intelligent Transportation Systems (ITS), smart cities, smart ecosystems, Internet of Things, and telecommunications. Several competing networking paradigms have been proposed to realize efficient vehicle to everything (vehicles, pedestrians, infrastructure, user devices, networks) applications. A good survey on the state of the art for intravehicular connectivity (eg. Control Area Network (CAN)) and extravehicular networking (eg. 5G) can be found in [89][71]. The automotive, as part of a rapidly evolving connected society, needs a semantic foundation to deal with heterogeneity in standards, communication protocols, data formats, application contexts and business players.

The semantic web was proposed by Tim Berners-Lee in 2001 as an extension of the World Wide Web that would link data to its subject matter similarly to how a web page links to another web page. These relationships can be expressed in RDF, an abstract model for semantic data as sentence-like statements about the world in triples of subject, predicate,

object. For example: the sentence "A cow" (subject) "is a" (predicate) "farm animal" (object), or "The mall parking lot" (subject) "has the number of free spaces" (predicate) "45" (object). As hinted at by these examples, triples can express both abstract information about classes (cows and farm animals) as well as facts about specific instances (the mall parking lot) and raw data values (45). A database designed and optimized for RDF data is known as a semantic repository or alternatively a triple store. The W3C SPARQL Protocol and RDF Query Language (SPARQL) recommendation [35] defines both a protocol and a query language for performing SQL-like operations on a semantic repository such as queries, insertions, updates, and deletes.

RDF is a natural way to express relational ontologies, as discussed in chapter 2, for semantic localization. Additionally, some semantic repositories, like GraphDB, support geospatial plugins for efficient queries over geocoordinates (i.e. lattitude and longitude pairs). If compatible with the GeoSPARQL standard [9], the semantic repository may also be able to automatically derive RCC (Region Connection Calculus) relationships, such as containment of one geospatial object within another, directly from the definitions of the objects themselves.

In this chapter we are interested in leveraging the semantic web stack toward facilitating smart thing interaction with a vehicle. The Semantic Sensor network ontology [19] and updated Sensors Observations, Samples, and Actuators (SOSA) ontology [42] codify domain knowledge regarding the abstract relationships of sensor network entities such as sensors, actuators, and the phenomena being sensed. When different semantic applications standardize on such an ontology, their knowledge bases become compatible in the sense that information may effortlessly be shared from one system to another. Pfisterer et al.'s Spitfire project coined the term "Semantic Web of Things" [76] to describe their approach in leveraging Linked Open Data [11] for the IoT. A good survey of progress to date in the Semantic Web of Things can be found in Barnaghi's report [8].

Researchers have applied semantic technologies to vehicles, but not to the best of our knowledge toward the goal of universal connectivity promised by the semantic web of things. W3C's Automotive Ontology working group[1] is developing shared vocabularies based on web ontologies for data interoperability in the automotive industry primarily for the purposes of standardizing vehicle information used for car rentals and sales. Schema.org now hosts these shared vocabularies at auto.schema.org as metadata markup for semantic web search.

Several ontology frameworks are being explored for use in autonomous vehicle technology such as Advanced Driver Assistance Systems (ADAS) to facilitate self-driving functions involving sensor actuation and control [5]. In [25], ontological knowledge systems were employed to interoperate with extra-vehicular infrastructure such as IoT devices or traffic lights to enhance traffic safety and driving experience. Our proposal is distinguished by our integration of semantic technologies with the accessors platform, designed to abstract away these communication difficulties. A connected car can just download an accessor (Section 4.3) for a sensor or actuator without worrying about the protocol or API the accessor uses

[1] https://www.w3.org/community/gao/

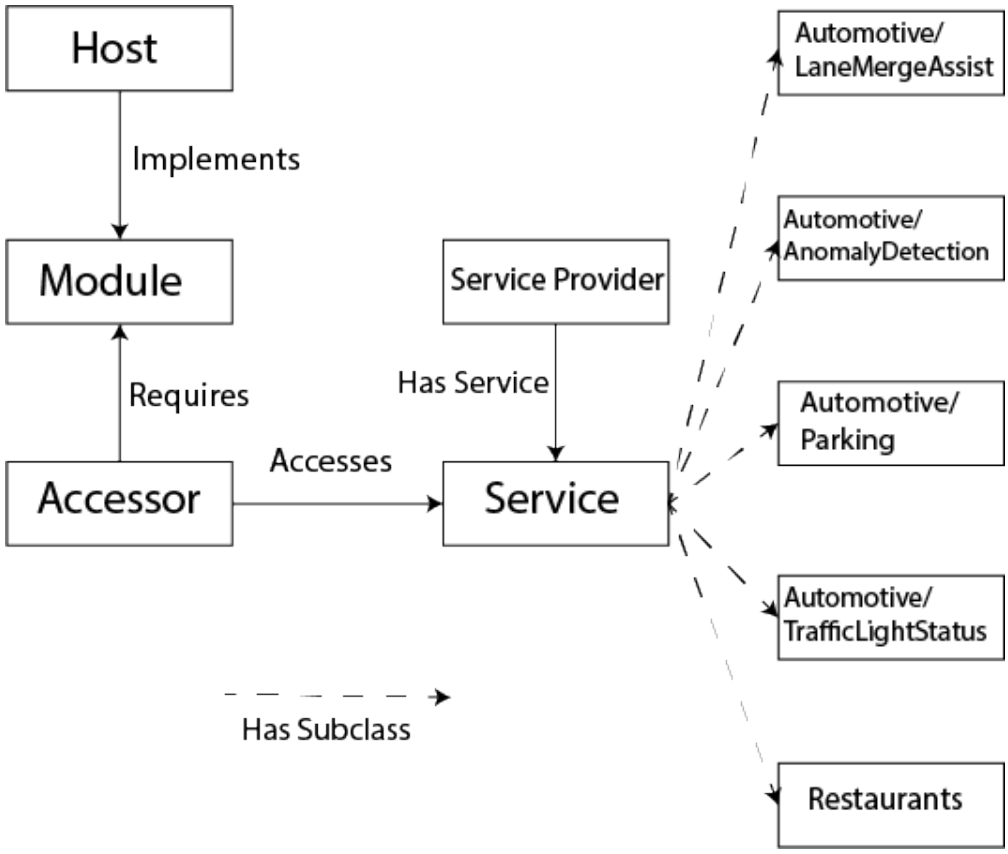

Figure 4.1: Diagram of accessor and service ontologies.

internally.

4.3 Semantic Accessors

In this section we describe our integration of semantic technologies with the Accessors project and present our work on new semantic accessors for connected vehicles.

Accessor and Service Ontology

The environment in which accessors and swarmlets execute is known as an accessor host (see chapter 3). As of this writing, the Accessors project has mature hosts that run on Node.js, in standard browsers, and in Java using Nashorn (Cape Code), as well as experimental hosts Cordova and Duktape for mobile and embedded devices respectively. To allow accessors to be platform independent, these hosts are responsible for providing native implementations for common environmental functions and modules. However not every host has the hardware to support every accessor. A Node.js host running in an intelligent vehicle may, for example,

not support the cameras module if the vehicle has no cameras. Before a newly discovered accessor can be selected, it is important to know whether this is the case.

To address this problem we developed an accessor ontology, a core excerpt[2] of which is presented in Fig. 4.1. We developed scripts that iterate through all known accessor and host libraries to populate the ontology with individual data for specific accessors, hosts, and modules. To be consistent with the principles of Linked Open Data [11], the name of an accessor in this generated ontology is a URI linking to a file containing its JavaScript implementation.

The ontology in Fig. 4.1 shows that an accessor may have an additional link to the service it accesses. For example, a parking service accessor ought to access a particular real-world parking service (an instance of Automotive/Parking) as the remote resource it proxies. Such a parking service is linked to its service provider through the Has Service relationship.

The combined Accessor and Service ontology is a powerful resource for semantic service discovery, and methods for its effective use are the core proposal of this paper. We present a scheme for managing these dynamic updates and service discovery in Section 4.4. It is worth noting these updates can include numeric, string, and other basic datatypes. It is a common misconception that ontologies cannot link basic data to concepts.

The script used to generate the accessor ontology works by instantiating each accessor from the accessor library in the Node host. Most properties of the accessor relevant for the ontology are obtained directly from the accessor object, with the exception of the "requires" relation. "Requires" is more difficult to obtain because the full list of module dependencies of an accessor may not be expressed directly in the code of the accessor itself. If the accessor is a composite it contains instances of other accessors and indirectly depends on modules needed by its sub-accessors. Similarly, an accessor which extends another accessor will execute code from its superclass accessor, potentially including calls to `require` from the baseclass.

As an additional difficulty, the Node host will throw an exception if an instantiated accessor attempts to call require on a module not supported by the Node host. This is desirable behavior for the Node host when an incompatible accessor is accidentally instantiated for execution at runtime, but is not desirable in a script designed to generate the accessors ontology for the entire accessors library which includes node-incompatible accessors.

To resolve these issues, we contributed a new function called `instantiateInterface` to the Node host. The function instantiates an accessor using an alternate version of `require` which 1) catches and ignores all exceptions from missing modules and 2) recursively logs the names of all modules required over the full instantiation process (which includes superclass and sub-accessors).

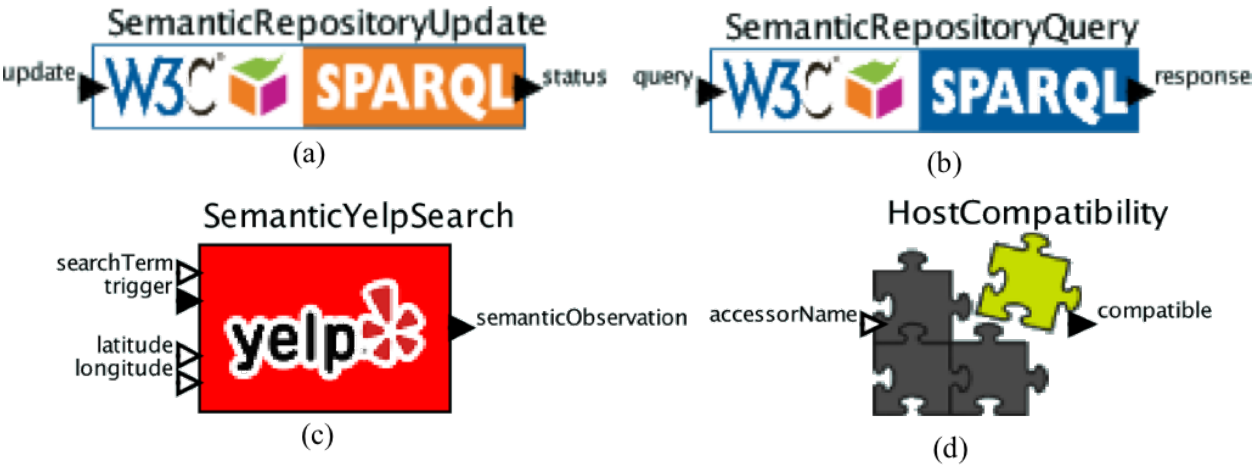

Figure 4.2: Actor interfaces to the accessors developed in this work

Semantic Accessors

In this research we have developed a new variety of accessor that interacts with semantic technologies: a semantic accessor. We illustrate the actor interface of four of these new accessors in Fig. 4.2 and elucidate their behavior in this section.

SemanticRepositoryUpdate and SemanticRepositoryQuery in Figs 4.2(a) and 4.2(b) proxy a SPARQL compatible semantic repository over HTTPS—it is irrelevant whether the repository runs locally or remotely. A SPARQL delete or insert command sent to the update port of the SemanticRepositoryUpdate will return with the success or failure of the operation on the accessor's status output. Similarly a SPARQL select, construct, ask, or describe query sent to the SemanticRepositoryQuery accessor will return with the query results on the accessor's response output.

Semantic repositories as conventionally used in the Semantic Web are relatively static entities used to maintain information in abstract domains such as medicine, government, linguistics, or media. We believe semantic repositories have enormous potential for use as a dynamic knowledge base for the immediate real-world context of a vehicle. Encapsulating the SPARQL protocol in these accessors integrates semantic repositories with the Accessor platform, where SemanticRepository Update and Query can be connected to other accessors in a swarmlet and used to bring the semantic repository into a dynamic control loop with sensors and actuators. An example of this use can be seen in figure 4.5.

The SemanticYelpSearch and HostCompatability accessors depicted in Figs. 4.2(c) and 4.2(d) represent more advanced semantic capabilities. When HostCompatability receives the name of an accessor as input, it checks the current state of the accessor's ontology (described

[2]The full accessor ontology is considerably more complex with concepts for Accessors, Accessor Interfaces, Inputs, Outputs, Parameters, Modules, Hosts, Services, and relationships for interface implementation, inheritance, and contained subaccessors among others. However, as the majority of these concepts are not related to service discovery, we have omitted them from Fig. 4.1.

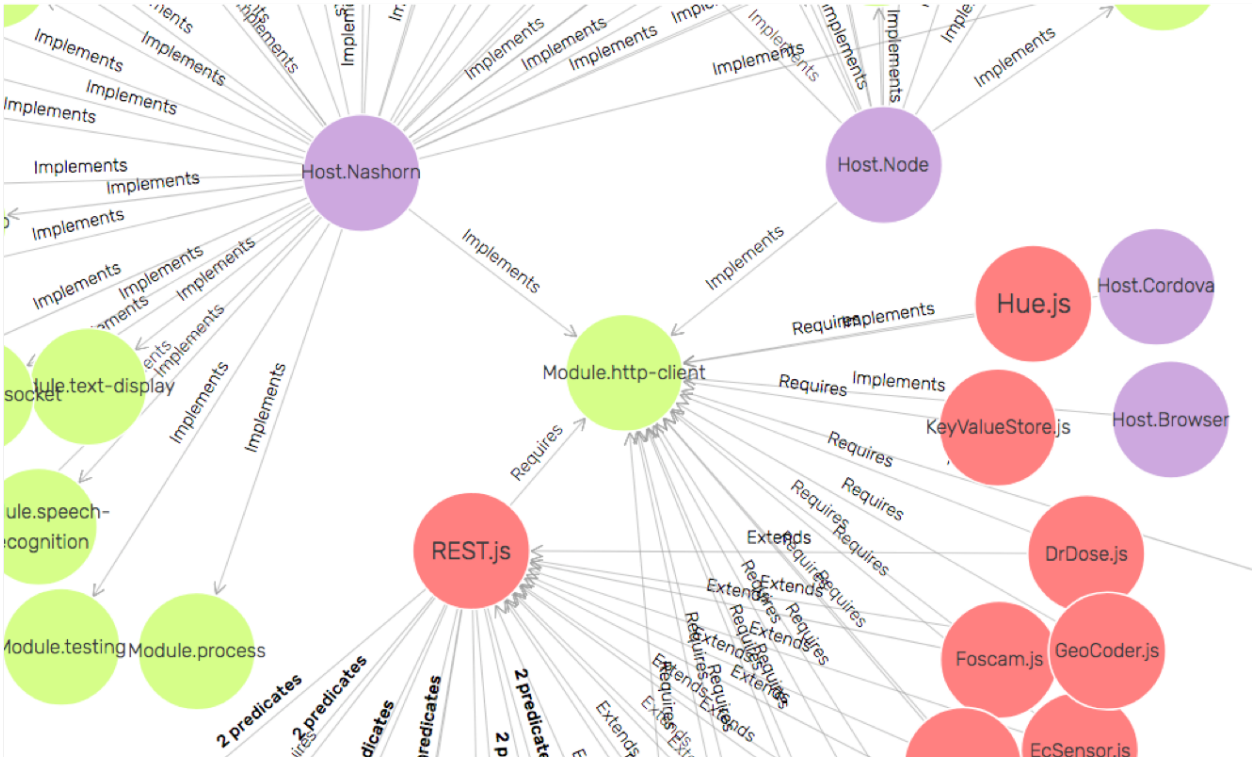

Figure 4.3: A zoomed in snapshot of objects in the accessors ontology. It can be seen the Nashorn and Node hosts both implement the http-client module, which is required by the REST accessor.

in Section 4.3 and depicted in Fig. 4.1) to determine whether the host it's running on is capable of providing that accessor's required modules. When the query is complete, the Boolean answer to the compatibility question is produced on its output.

A snapshot of the accessor's ontology is given in Fig. 4.3. As can be seen from the snapshot, the REST accessor requires the http-client module, which is implemented by both the Nashorn host and the Node host (among others). The HostCompatability accessor works by querying this ontology for the list of modules supported by the current host and compares them against the list of modules required by the given accessor. Therefore it can be determined from this snapshot the REST accessor is compatible with the Nashorn and Node hosts.

Some services may be known in advance and hence statically encoded into the ontology before deployment, but others may be dynamically discovered (for example, nearby Restaurant services via SemanticYelpSearch) and inserted into the ontology with SemanticRepositoryUpdate.

The SemanticYelpSearch accessor is an example of a general class of *semantic sensor* accessors. These accessors produce as output semantic observations in the form of Turtle syntax ontologies. The name "semantic sensor" is not an accident: the output ontology uses

concepts from the W3C SOSA[3] ontology for sensor observations. SemanticYelpSearch takes a location and search topic as input and when triggered, interacts with the Yelp API to produce as output a semantic observation of nearby local businesses.

Since the Yelp API does not use a semantic representation for its results, SemanticYelpSearch uses the N3 library to translate Yelp results into schema.org's LocalBusiness ontology. In fact, the ease of *encapsulating an arbitrary non-semantic API with an accessor to translate its results into a semantic output* is one of the chief advantages of semantic sensor accessors. Mapping data to standard SOSA and schema.org ontologies facilitates automatic integration of data obtained from a semantic sensor accessor with any other SOSA and schema.org compatible ontologies, such as the service ontology depicted in Fig. 4.1.

4.4 Semantic Accessor Architecture

In this section we present an architecture for semantic accessors that enables semantic service discovery and a dynamic user interface for service interaction. The main system components of this approach are diagrammed in Fig. 4.4, and the swarmlet controller that coordinates these components is given in Fig. 4.5. The controller has four interconnected parts: the semantic repository maintenance swarmlet, which manages information about the world; a service selection user dialogue, which determines user intent for service interaction; a mutable controller for web component accessors, which downloads and sets up communication with the selected service; and an accessor for the dynamic user interface itself. As shown in Fig. 4.4, the controller and semantic repository run locally on the vehicle and display a user interface on the vehicular dashboard. SemanticSensor accessors in the controller proxy remote services in the cloud such as Yelp, and also nearby fog resources such as traffic lights and roadside units. When service discovery is complete, the user can interact with cloud and fog resources (eg., paying for parking) through UIComponent accessors (Section 4.4).

Swarmlet Controller

We address the aspects of the swarmlet controller[4] depicted in Fig. 4.5.

Semantic Repository Maintenance

Knowledge in the vehicle's local semantic repository comes from two sources: the static core ontology initialized into the repository with basic information about service concepts (or universally relevant services) and the contextually relevant dynamic ontology provided by SemanticSensor accessors. Directed by user interest from the dashboard, the semantic repository maintenance swarmlet periodically acquires new location-specific information

[3]SOSA is a rethinking of the classic *Semantic Sensor Network* ontology.

[4]For clarity of presentation, this diagram omits slots for multiple semantic sensors, multiple service selection dialogues, multiple downloaded accessors, and the message routing system which ensures in-swarmlet communication is directed to the correct slot.

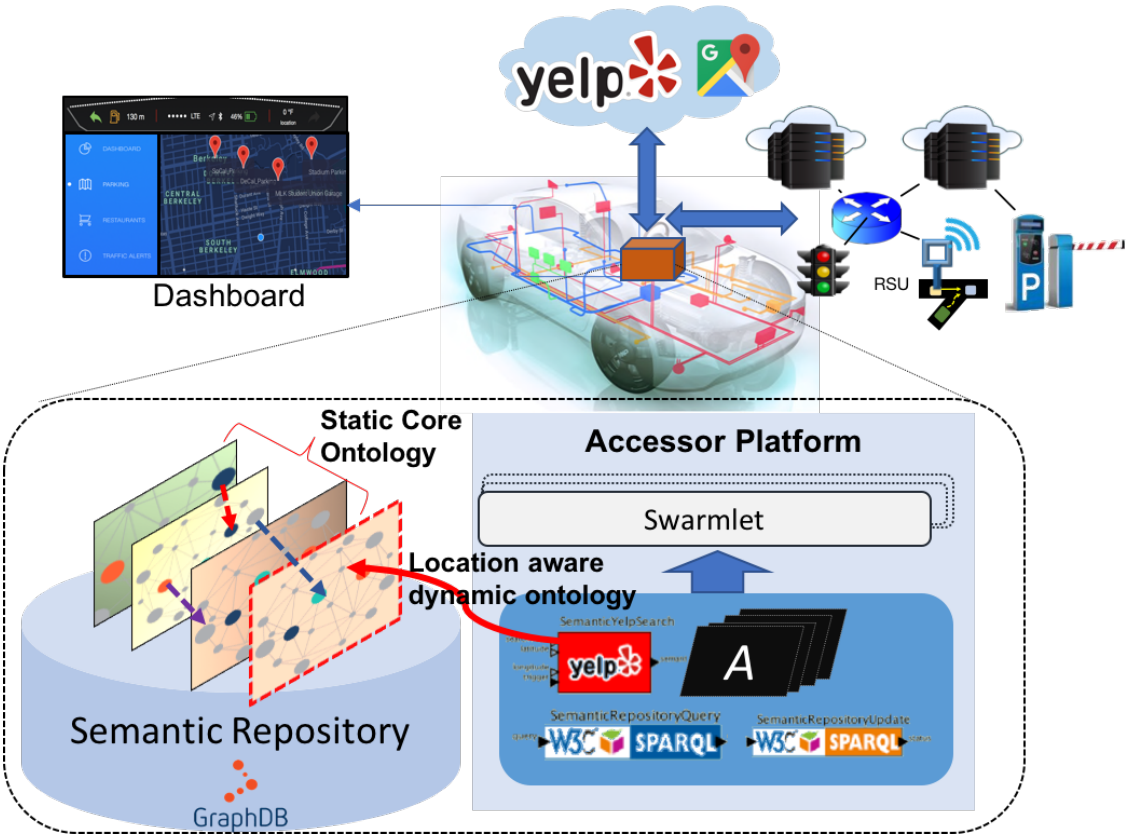

Figure 4.4: Elements of the Semantic Accessor Architecture.

from semantic sensors (such as SemanticYelpSearch) and writes the data into the dynamic ontology. This use of a semantic repository is analogous to a streaming database like Apache Kafka or AWS Kinesis but comes with the advantages discussed at length in sections 4.2 and 4.3.

To avoid endless maintenance of stale data in the SemanticRepository,[5] a SPARQL delete command also runs periodically to delete old semantic observations. A semantic repository that supports the GeoSPARQL standard for geographic data (like GraphDB) can also be set up with a SPARQL delete command to periodically delete old observations by geofencing and deleting observations spatially located far from the vicinity of the vehicle.

Service Selection Dialogue

With the semantic repository kept fresh with contextually relevant data, the controller is ready to respond to user requests for service discovery. In response to a service topic from the user interface (eg. parking services), the service selection dialogue finds relevant host-

[5]Plus, it is against the terms of Yelp's API usage to store their data for more than 24 hours and "build another Yelp". The same is likely true with other service providers.

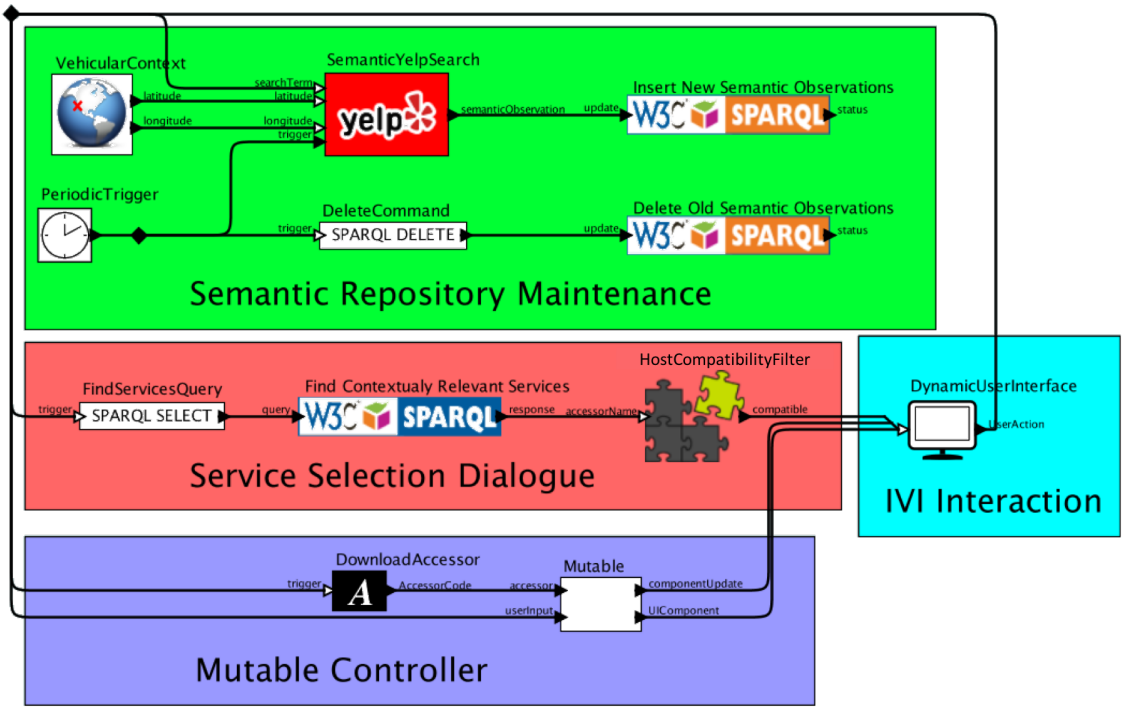

Figure 4.5: Illustration of the Semantic Accessor Architecture's swarmlet controller

compatible services using the SemanticRepositoryQuery and HostCompatability accessors respectively. We developed a prototype user interface for the resulting IVI display, shown in Fig. 4.6. The point isn't to recreate Yelp or Google Maps, but to present the user a fusion of contextual information from the repository they couldn't get from any individual cloud service.

Mutable Controller and Dynamic User Interface

Once a service is identified by the service selection dialogue, its accessor is downloaded by the mutable controller and sent to a mutable accessor. A mutable accessor is a special higher-order accessor, presented in [15], which can be thought of as an open hole in the swarmlet that fills itself with the accessor code it receives on its "accessor" input. In this way the downloaded accessor is *reified* (i.e. made real and plugged into the model) in place of the mutable, as shown in Fig. 4.7.

This particular mutable accessor requires that the accessor it reifies for a connected car service implement the UIComponent accessor interface. Upon reification in a mutable, a UIComponent produces a definition for an interactive user interface component on its UIComponent output in the form of a web component,[6] which will be rendered as part of

[6]Web components are a newly released standard for creating encapsulated bundles of HTML, CSS, and JavaScript, which are reusable as interface components across web apps. For more information see:

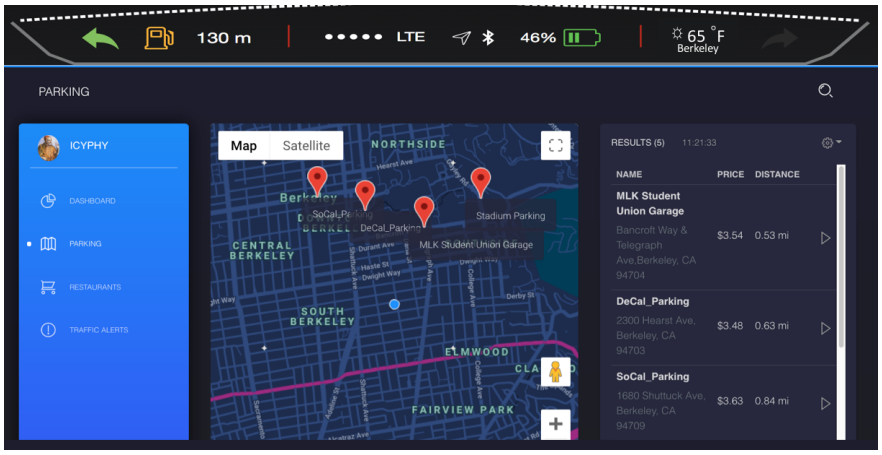

Figure 4.6: Screen capture of our user interface for a parking selection dialogue. Parking options are displayed in both a Google maps component and a table sorted by price or distance. Selecting a marker in the map or a row of the table triggers the mutable controller to download the accessor for that particular parking service. Marker locations and distance are dynamically generated from the semantic repository.

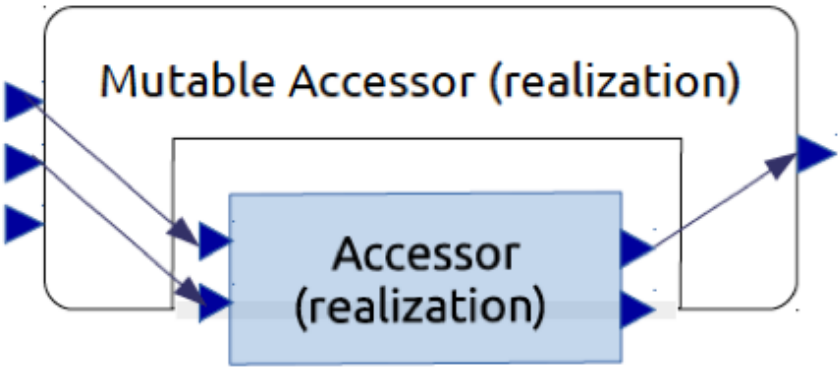

Figure 4.7: A mutable's behavior can change during the lifetime of a swarmlet. Image courtesy of Chadlia Jerad.

the user interface. In this way a UIComponent is self-describing!

Our prototype of the dynamic user interface is a React.js app.[7] The prototype communicates with the swarmlet controller over web socket. Web apps have been rendered in IVIs (for example, using application frameworks provided by Automotive Grade Linux (AGL)). When the React app receives a web component from the mutable controller, the component is rendered as an interactive card, as shown in Fig. 4.8 for a parking component. User interaction with such a card is directed back to the mutable controller and the Mutable's

https://www.webcomponents.org/introduction.

[7]Built off the Black Dashboard React template by Creative Tim (MIT Licensed)

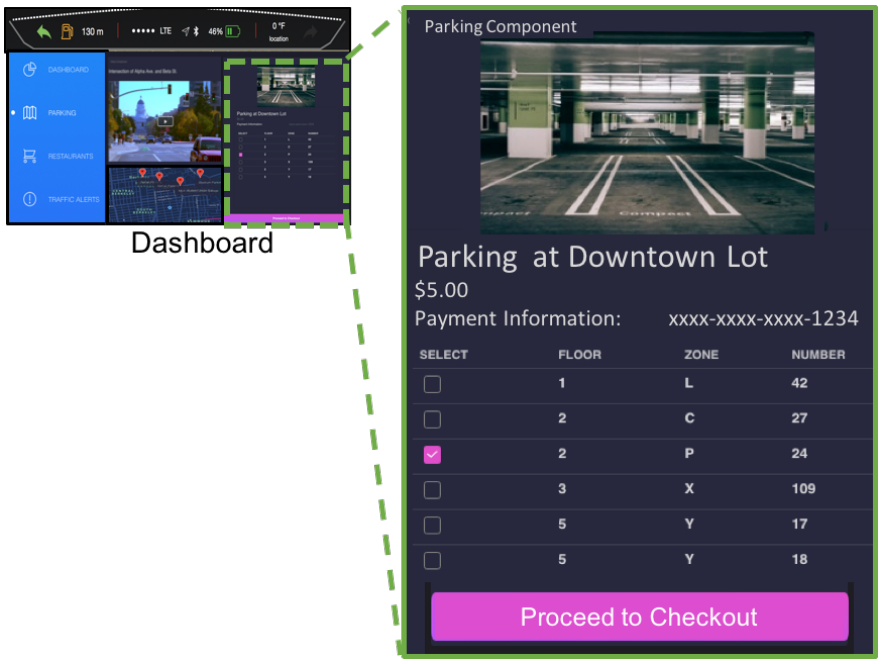

Figure 4.8: Prototype implementation of a user interface web component for an example parking service. This web component, and our other prototype components were built with React.

userInput port, providing the reified accessor (and the service it proxies) the opportunity to respond to user interface events. The reified accessor may update the UI component in the React app by producing an output over componentUpdate.

For example when a parking service is selected from the parking dialogue in Fig. 4.6, the selected parking accessor is downloaded and reified in the mutable. It produces the parking web component shown in Fig. 4.8 on its UIComponent output. When the parking component is first rendered in the React app, it doesn't know about the current status of parking spots so it initiates communication back to the mutable controller and the refied parking accessor's userInput port. The parking accessor communicates with the parking service it proxies to obtain this information and sends it back to the instantiated component via the componentUpdate output. The parking component receives the data and renders the current parking information. Similar to the parking component we also prototyped a video component intended to show a clear view of traffic from an intersection when the driver's view is obstructed, and a restaurant menu component for ordering from a fictional restaurant.

Dashboard in Cordova

So far we have discussed the React UI as an automobile-compatible webapp. However the same dynamic UI paradigm and swarmlet controller also works without substantive changes

as a Cordova Swarmlet. Recall from section 3.2 that the default Cordova UI consists of only just a simple display of swarmlet state information and a text console. We have demonstrated a proof of concept for the dashboard UI as an alternative Cordova UI. This is accomplished by replacing the default index.html for the Cordova UI with one which loads the React UI along with the swarmlet controller.

There were some noteworthy challenges in building the proof of concept for this alternative interface. Since the React UI communicates with the swarmlet controller through websocket to determine which web components to render, the UI cannot be started until after the swarmlet controller has completed setting up. We addressed this by adding a cordova-specific "UI" accessor to the controller which causes the React UI to be loaded in index.html when the UI accessor receives a "ready" message from the DynamicUserInterface accessor.

As another challenge, we had to make special modifications to index.html's Content Security Policy (CSP). CSPs are commonly used by browsers to protect trusted web pages from cross-site scripting attacks in which malicious foreign code is allowed to execute within the trusted page. A subtle CSP issue arises when the dashboard is architected as a Cordova application: if the swarmlet controller and React UI both execute within the same webview, from the webview's perspective downloaded accessors and the web components they create are foreign code. Therefore to complete our implementation of the Cordova dashboard, we had to modify index.html's CSP to a very permissive setting, allowing the evaluation of arbitrary scripts, images, stylesheets, and fonts.[8] As this change leaves the Cordova dashboard vulnerable to actual cross-site scripting attacks, we believe it will be necessary in the future to implement an accessor whitelist as a custom security mechanism to ensure only trusted accessors and web components are allowed to be loaded.

4.5 Conclusion

From traffic infrastructure that helps you make the light from a mile away and avoid an accident, to restaurants that let you order before you arrive and save you a parking spot, the future holds exciting possibilities for connected cars. But for these capabilities to be fully realized, vehicles must acquire dynamic knowledge of their world. They must learn about connected services in the environment, how the services relate to each other, and how to interface with a service's intelligent capabilities. In this chapter we proposed Semantic Accessors as a fusion of the technologies of the Semantic Web with the IoT interaction of the Accessors project. We presented an ontology for accessors and vehicular services that enables the HostCompatability accessor to enhance swarmlets with the power of semantics and semantic sensors like SemanticYelpSearch to bring the flexibility and composition potential of accessors to bear on semantic repositories. The Semantic Accessor Architecture we prototyped alongside a dynamic user interface for Dashboard displays demonstrates how these components can be combined to build flexible and contextually intelligent applications.

[8]Module bundlers like webpack (used to compile the React UI) encode images within fonts.

In future work, we intend to investigate logical and machine learning inference to enhance the knowledge stored in a semantic repository. For example, a faulty street address for a restaurant might be identified by inferring a discrepancy between the advertised geolocation, street address, and geolocation where the vehicle actually parks. It would also be interesting to enhance the semantic repository maintenance swarmlet with mutable slots for discovered semantic sensors. In this way, a semantic repository might accumulate data sources as it progressively infers newly relevant sensors. And finally, enhancing our dynamic interface prototype with security and authentication methods from Kim's locally centralized, globally distributed authentication agents is a prudent avenue for future investigation [45].

Chapter 5

Adapters for Dynamic IoT Services

As stated in the chapter 1, a **dynamic IoT service** *opportunistically discovers contextually relevant devices and combines them into new IoT applications.* Where chapter 4 presented an approach for building dynamic IoT services designed for human control via the dashboard interface, in this chapter we address additional challenges in building dynamic IoT services without a human in the loop. The goal of this work is to enable IoT services to automatically interact with newly discovered sensors and systems on the fly. In particular we address the problem of semantic interpretation between systems. Where accessors are a solution for the challenges of heterogenous communication protocols and data types, we define **semantic interpretation** as *the task of interpreting received data as actionable information for the control of a system* and propose as a solution a new kind of accessor we call an adapter.

Systems with a user interface can rely on human understanding when interacting with a new device. A human can open a menu and read the available options. A good user interface is intuitive and built upon a common language for human computer interaction that people who have used similar interfaces in the past can easily interpret. For example most smartphone users don't need to be told to tap on a user interface element to select it on a touch screen. In 2019, they can figure that out on their own.

However when sensors and systems interact with each other directly in a dynamic IoT service, they must semantically interpret each other's interfaces without relying upon human intelligence. There are several possible solutions: In static IoT services a sensor's interface may be specifically engineered for a particular system (or vice versa) so it can know in advance how to interpret the data it receives. Similarly, if the sensor and system agree in advance to use a particular standard for data interpretation, they may semantically interpret each other without difficulty. But as argued in chapter 2, contextual/spatial models of the world are diverse and appropriate for different applications. Not only is universal standardization difficult, it can be challenging to reconcile data across contextual models. The adapters we propose in this chapter are a solution for this third scenario where sensors and systems may be discovered on the fly and must then discover a way to understand each other.

The main contributions of this chapter are:

- A swarmlet design pattern for dynamic IoT services featuring adapters: an accessor component for semantic interpretation
- A proof of concept demonstration of the above with a robot following assistant
- A model theoretic formalization of ontology matching with adapters and a proof that adapters may be used as part of a valid inference procedure on ontologies
- Meta-ontology infrastructure for discovering adapters using semantic accessors from chapter 4

This chapter presents a synthesis of the preceding chapters, combining the contextual modeling of chapter 2, the component architecture of chapter 3, and the semantic accessor architecture from chapter 4 to address semantic interpretation in dynamic IoT services. We begin in section 5.1 by introducing a swarmlet design pattern for dynamic IoT services as motivated by our implementation of a robot following assistant, continue in section 5.2 with a formalization of adapters as a form of ontology matching, and give a meta-ontology infrastructure needed to automate discovery and reification of adapters in section 5.3, and conclude in section 5.4 with a discussion of related and future work.

5.1 Designing Dynamic IoT Services

Consider a general IoT service which integrates sensor data with a system. When data becomes available from the sensor, the system acts upon that data by performing a desired task (actuation, analysis, etc.). A static IoT service can be designed by hardcoding a particular sensor to interact with a particular system across a predefined communication link. However a dynamic IoT service must be more flexible to allow an arbitrary sensor to interact with an arbitrary system. We propose a pattern for building dynamic IoT services, illustrated in figure 5.1, in which the service is defined as a swarmlet indicating the connections for discovered accessors for a sensor and system. Such a swarmlet is more of a specification for service behavior than an implementation because it has a mutable sensor and a mutable system. It becomes executable[1] when the accessor for a particular sensor and a particular system are reified in the swarmlet.

As a concrete example of a dynamic IoT service fitting this pattern, consider a robot following assistant. In this scenario a robot is intended to follow a human as they move through an indoor environment in order to aid the human in some task. The robot can localize itself in space, but doesn't come with all the sensors it needs to localize the human its supposed to follow. Fortunately, the human's environment is equipped with a human-tracking indoor localization system and is capable of providing the missing information to the robot. A swarmlet for the robot following assistant service matching the pattern of

[1]Figure 5.1 leaves out most of the details on how a contextually relevant and host compatible accessor can be selected from a semantic repository and accessor library. Refer to chapter 4 for an in-depth discussion.

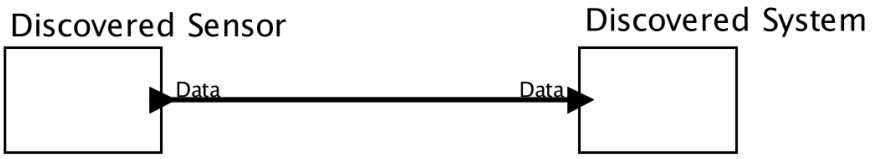

Figure 5.1: Illustration of a (generalized and simplified) swarmlet specifying a dynamic IoT service. Both the Discovered Sensor and Discovered System accessors are mutable.

figure 5.1 can be designed by discovering an accessor for the human tracking (sensor) and discovering an accessor for an available robotic assistant (system).

We implemented accessors for a sensor and a system matching these descriptions using out-of-the-box components. The sensor was set up using the open source Framework for Internal Navigation and Discovery (FIND)[2] WiFi fingerprinting-based indoor localization system. FIND has two parts: a mobile app client and a central server. To configure an instance of FIND for a new location, a human has to walk around the place of interest with their phone and perform WiFi and bluetooth signal strength scans using the mobile app in learning mode. These samples of data are labeled by the human with the name of the location where the scan took place (eg. room 1, room 2, the kitchen, or the couch) and sent to the central FIND server. The server trains a suite of machine learning classifiers with the data including random forest, support vector machine, and AdaBoost. Later, when a user enables the FIND client in tracking mode and sends unlabeled scans to the server, the classifiers are used to estimate the likely positions of the user.

The robot assistant system was set up using a Turtlebot 3 Waffle.[3] Like FIND, the Turtlebot comes out-of-the-box with software capable of performing navigation, but requires some setup to work in a particular space. The Turtlebot comes equipped with ROS (Robot Operating System) packages useful for localization, mapping, and navigation. The Turtlebot must first learn a map of its environment. After launching ROS nodes for Simultaneous Localization and Mapping (SLAM), the robot will build its map using a laser distance scanner as a human drives it around the desired location. Once parameters for navigation, including the dimensions of the robot, movement speeds, and aggressiveness in tight spaces are set, a ROS navigation node may be launched to automatically navigate the robot to a desired location published on its subscribed ROS topic.

We implemented accessors for both FIND and the Turtlebot. The data produced by the FIND accessor is the location of the human and the data consumed by the Turtlebot accessor is the location where the robot should go. The FIND accessor uses the FIND server's REST

[2]Available at www.internalpositioning.com

[3]The same robot as depicted in figure 3.9

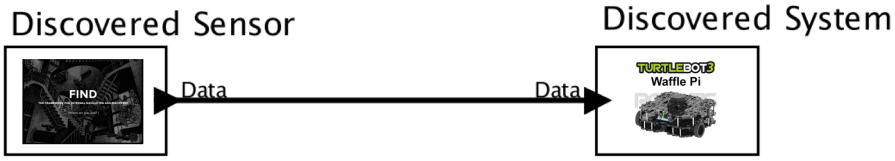

Figure 5.2: The dynamic IoT swarmlet reified with the FIND location sensor and a Turtlebot 3 Waffle system. Images inside the boxes courtesy of www.robotis.us/turtlebot-3-waffle-pi and www.internalpositioning.com.

(HTTP) API to obtain current tracking data for the specified user. The Turtlebot accessor uses a websocket connection to an instance of the Rosbridge node running on the robot. Rosbridge publishes the received message to the navigation topic, initiating movement to the designated map position. After reification the dynamic swarmlet looks like figure 5.2.

There is however a subtle problem with figure 5.2 an astute reader may have anticipated from the discussion of semantic interpretation: the location representations of the two systems are initially incompatible. FIND obtains a probability distribution over the rooms most likely to contain the user. The screenshot of such a location result from the FIND dashboard is depicted in figure 5.3. The robot however navigates over a ROS occupancy grid of its surroundings, which is essentially a large 2D array of cells representing the probability of an obstacle at a location A depiction of the occupancy grid map of part of the DOP Center (our lab) is given in figure 5.4.

We propose a solution to semantic incompatibly by way of a modification to the dynamic IoT service swarmlet. This change, depicted in figure 5.5 is the addition of a new mutable component to the swarmlet between the sensor and system called an adapter. The adapter's job is to convert data from the discovered sensor's spatial/contextual model (in the case of FIND, room labels) into something semantically interpretable within the discovered system's spatial/contextual model (i.e. occupancy grid coordinates for the Turtlebot). To accomplish this, the reified adapter implements an ontology matching function. The result is the working robot following assistant shown in figure 5.6, which converts FIND room labels such as DOP Couch to Turtlebot navigation goals.

This demo illustrates the essential tasks in designing a dynamic IoT Service: sensor discovery, system discovery, and adapter discovery. We propose addressing discovery for sensors and systems in the same manner as chapter 4: a semantic repository with contextual information about the world is maintained alongside an accessors ontology, and is queried to obtain the most appropriate accessor for the sensor and system. The swarmlet in figure 5.5 is both a specification for these tasks, and a template for how data should be directed between the discovered accessors. The result is a user interface-free variant of the semantic accessor architecture.

However, the need for semantic interpretation across systems demands additional infras-

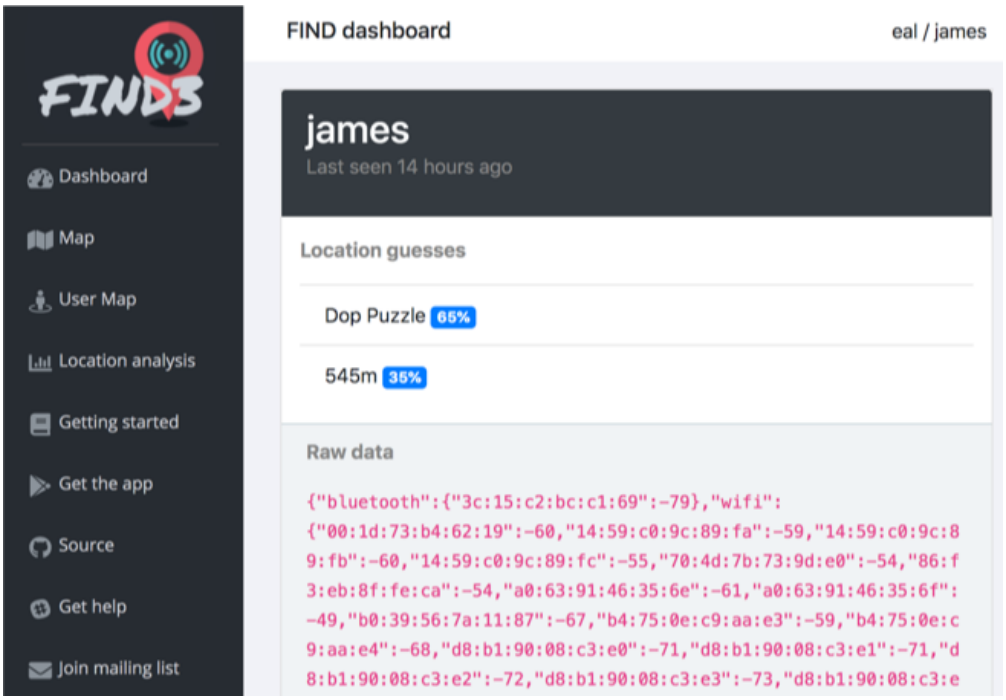

Figure 5.3: Screenshot of location estimation results from the FIND dashboard.

tructure for the adapter. We elaborate on the extension of the semantic accessor architecture needed to automate the discovery and reification of adapters in section 5.3. But we will first address in section 5.2 a mathematical description of what exactly an adapter does, and what it takes for it to perform a correct and useful matching between spatial/contextual ontologies.

5.2 Ontology Matching and the Theory of Adapters

Recall from chapter 2 the notion of a spatial/contextual ontology as a particular model of the world. Figure 5.2 was problematic because the data produced by the sensor is a concept from the sensor's ontology and the data expected by the system is a concept from the system's ontology. Adapters have a strong connection to a topic in pure ontological research known as ontology matching[4] in which the correspondences between concepts in different ontologies are determined. In this section we give a formalization of adapters as ontology matching for the spatial/contextual models of chapter 2. Adapters draw upon ideas from a wide variety of research areas. Ideas from ontological matching research play a prominent role, as do

[4]Depending on whose terminology you use there are distinctions between the related tasks of "ontology matching", "ontology mapping", and "ontology alignment". According to Godugula [31] "matching" is the task of indicating related concepts across ontologies (aligning) while "mapping" is the similar task for database schemas. However Shvaiko and Euzenat [88] use the term "matching" with respect to ontologies. We have adopted Shvaiko and Euzenat's usage.

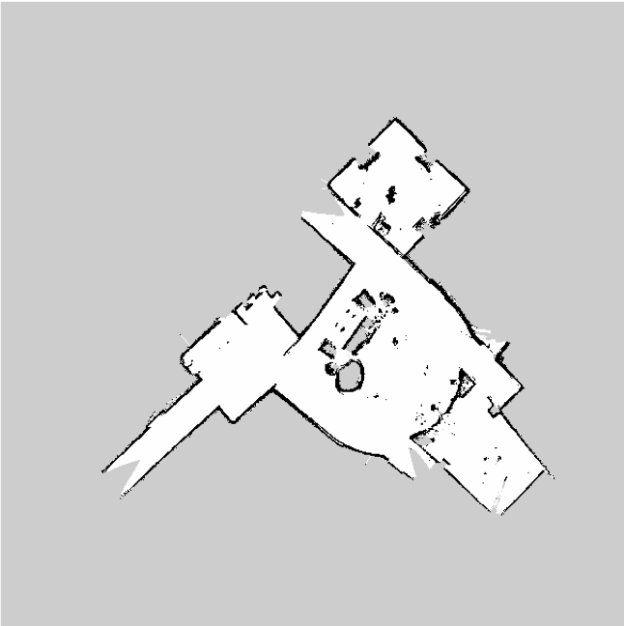

Figure 5.4: ROS occupancy grid map of the DOP center, obtained from the Turtlebot.

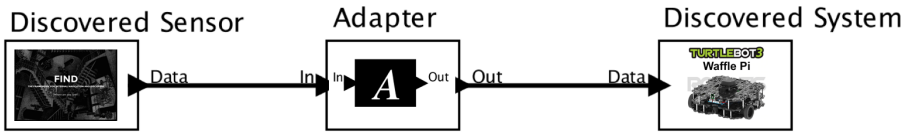

Figure 5.5: An adapter has been added to the dynamic IoT swarmlet to address semantic incompatibility between the sensor and the system.

conceptualizations of maps as mathematical objects. There are also relations to ideas from the world of AI and robotics where researchers have very practical motivations for developing spatial ontologies.

Related Work

Surveys on the topic of ontology matching [31, 88] discuss research steps toward automatically building matchings between ontologies through a variety of methods such as graph analysis and word similarity of terms. As an example of graph analysis, one way of evaluating the quality of automatically generated matchings is to determine if ontological relationships of an object to other objects in its ontology are maintained through the matching. This avenue of ontological research is usually applied outside of a spatial context.

An algorithm for deductive ontology translation is given in [20], where the distinction

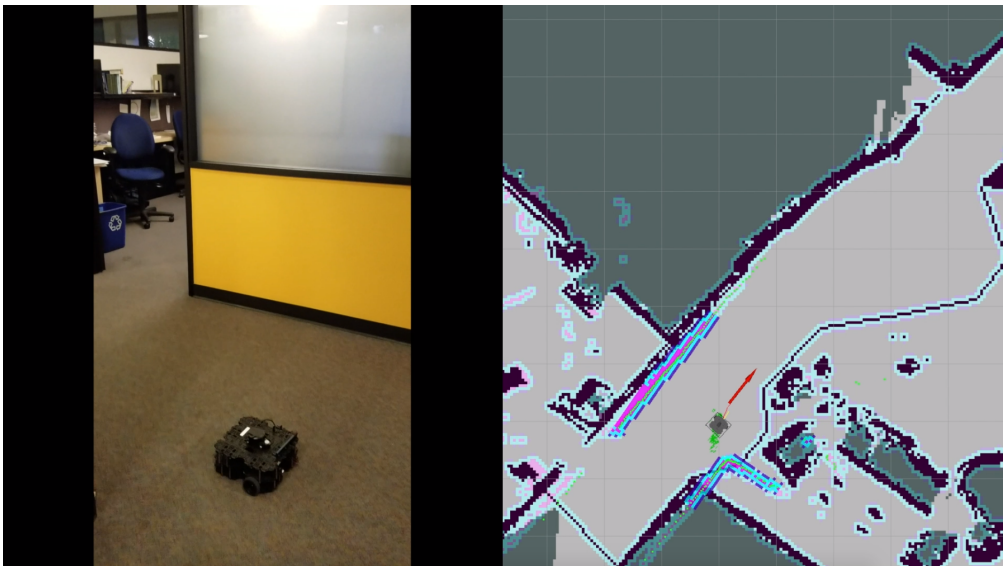

Figure 5.6: The robot following assistant in action. Video footage (left) of the Turtlebot making its way to the DOP Center couch, alongside a visualization (rviz) of the Turtlebot's navigation on the occupancy grid. The robot's goal position on the grid is represented by the red arrow, and colored areas indicate high cost (dangerous) areas for the robot to avoid in navigation. The green dots are the robot's location estimates for itself.

between ontology matching and translation is stressed. Essentially, translation is the modification of a dataset for a particular ontology such that it can be understood in the context of a different ontology.

Motivated primarily by the semantic web, Spaccapietra et al. [91] seek to bridge the gap between geographical ontologies and "conceptual data models" developed by the database community. The authors survey three kinds of logic based models for ontologies: Description logics, F-Logic, and Horn-Logic. After describing alternative aproaches for the application of these to spatial ontologies, they propose their own spatio-temporal data model: MADS. Similarly, Fonseca et al. [26] attempt to link formal semantics expressed in ontologies for spaces to conceptual schemas found in databases. Their approach gives a formal framework for translating between ontologies and conceptual schemas and is ultimately used to describe a work flow for building a geospatial application.

In a similar vein to our proposal in this chapter, Câmara et al. [17] give an algebraic definition for spatial ontologies with the goal of mathematically formalizing equivalence and transformations on ontologies. The main advantage of their method is the idea expressed in their proposition 3: "The set of ontologies constitute an Abelian monoid" under an on ontology composition operator. Their formalism allows maps to be merged side-by-side, i.e. given two maps the fusion of the information expressed in each into a third combination map. But this approach effectively discards the cross-ontology matching after performing a merge, and we will argue in section 5.3, matchings are useful objects in their own right.

A very practical application of ontology matching with linear transformations between Euclidean models of space is implemented by the *tf* ROS library [27]. Controlling a robot frequently involves programming with respect to different Coordinate Reference Frames (CRF). *tf* uses a tree data structure (each CRF is attached to a parent) to respond to asynchronous requests for transformations from one reference frame to another. In future work we will investigate whether the *tf* approach may be leveraged for more sophisticated matchings as well.

Model Theory for Ontology Matching

Recall our proposal for semantic localization from section 2.2 where we defined a spatial/contextual ontology as a model theoretic structure. As model theory also provides vocabulary for describing the connections between related structures, we introduce that terminology here and use it below to formalize ontology matching for adapters. As in section 2.2, we use [102] as a reference. Again, this is a quick overview and not a full model theory tutorial.

Structures and languages may be related to each other through the addition or removal of symbols. Consider a language $\mathbb{L}$. A language $\mathbb{L}'$ is an **expansion** of $\mathbb{L}$ iff $\mathbb{L} \subseteq \mathbb{L}'$. In this case $\mathbb{L}$ is a **reduction** of $\mathbb{L}'$. If $\mathbb{A}$ were a model of $\mathbb{L}$, we could create $\mathbb{A}'$, a model of $\mathbb{L}'$, by giving $\mathbb{A}$ interpretations for the symbols of $\mathbb{L}' \setminus \mathbb{L}$ (i.e. all the new symbols of the expansion language). In this circumstance $\mathbb{A}'$ is an **expansion** of $\mathbb{A}$ to $\mathbb{L}'$ and $\mathbb{A}$ is the **reduct** of $\mathbb{A}'$ to $\mathbb{L}$. This reduct is denoted as $\mathbb{A}'|\mathbb{L}$.

Similarly, structures may be related to each other by the addition or removal of domain elements. Consider model $\mathbb{A}$ with domain $\mathbf{A}$. The model $\mathbb{A}'$ with non-empty domain $\mathbf{A}'$ is a **submodel** of $\mathbb{A}$ (denoted $\mathbb{A}' \subseteq \mathbb{A}$) iff $\mathbf{A}' \subseteq \mathbf{A}$ and the following conditions are true with respect to the constants, functions, and relations of $\mathbf{A}'$:

- All constants of $\mathbb{A}'$ are corresponding constants of $\mathbb{A}$.
- All n-placed functions of $\mathbb{A}'$ are corresponding functions of $\mathbb{A}$ but restricted to $(\mathbf{A}')^n$
- All m-placed relations of $\mathbb{A}'$ are corresponding relations of $\mathbb{A}$ but restricted to $(\mathbf{A}')^m$

An **elementary submodel** is a special kind of submodel in which both models maintain logical properties. $\mathbb{A}$ is an elementary submodel of $\mathbb{B}$ (denoted $\mathbb{A} \prec \mathbb{B}$) iff structures $\mathbb{A}$ and $\mathbb{B}$ are both models for language $\mathbb{L}$, $\mathbf{A} \subseteq \mathbf{B}$ and all formula ϕ of $\mathbb{L}$ with variable assignments to free variables ranging over $\mathbf{A}$ (i.e. $\phi(a_0, a_1, ..., a_k)$) satisfies $\mathbb{A} \models \phi(a_0, a_1, ..., a_k)$ iff $\mathbb{B} \models \phi(a_0, a_1, ..., a_k)$. In other words when $\mathbb{A} \preceq \mathbb{B}$, even though the domain of $\mathbb{A}$ is a subset of the domain of $\mathbb{B}$, $\mathbb{A}$ and $\mathbb{B}$ never disagree on the truth value of any formula with variables from $\mathbf{A}$.

Model theory has several notions of model equivalence, but we will focus on isomorphism. Two models $\mathbb{A}$ and $\mathbb{A}''$ with the shared language $\mathbb{L}$ are **isomorphic** (denoted $\mathbb{A} \cong \mathbb{A}'$) iff there exists a bijection $f : \mathbf{A} \to \mathbf{A}'$ satisfying:

- Each constant symbol of $\mathbb{L}$ corresponding to a in $\mathbb{A}$ and a' in $\mathbb{A}'$ satisfies $f(a) = a'$.
- All n-placed functions g of $\mathbb{A}$ and g' of $\mathbb{A}'$ for all $x_1, x_2, ..., x_n \in \mathbf{A}$ satisfy $f(g(x_1, x_2, ..., x_n)) = g'(f(x_1), f(x_2), ..., f(x_n))$.
- All n-placed relation symbols r of $\mathbb{A}$ and r' of $\mathbb{A}'$ for all $x_1, x_2, ..., x_n \in \mathbf{A}$ satisfy $r(x_1, x_2, ..., x_n) \leftrightarrow r'(f(x_1), f(x_2), ..., f(x_n))$

We finally have the vocabulary to introduce the model relations most important to ontology matching: embeddings. Model $\mathbb{A}$ has an **isomorphic embedding** into model $\mathbb{B}$ iff there exists a model $\mathbb{C}$ satisfying $\mathbb{A} \subseteq \mathbb{C}$ and $\mathbb{C} \cong \mathbb{B}$. $\mathbb{A}$ has an **elementary embedding** into $\mathbb{B}$ iff there exists a $\mathbb{C}$ satisfying $\mathbb{A} \prec \mathbb{C}$ and $\mathbb{C} \cong \mathbb{B}$. These embeddings capture desirable qualities for matchings between spatial ontologies. We will discuss below how isomorphic embeddings enable cross-ontology inference and how elementary embeddings facilitate composition of ontologies for semantic localization.

Ontological Inference with Matchings

Consider an adapter which implements a function f relating the contextual models of a sensor and a system. Let the sensor's contextual model be structure $\mathbb{A}$ and let the system's contextual model be structure $\mathbb{B}$. For the adapter to perform a semantic translation, as in figure 5.5, f must be a matching from the domain of $\mathbb{A}$ to the domain of $\mathbb{B}$ ($f : \mathbf{A} \to \mathbf{B}$).

But $\mathbb{A}$ and $\mathbb{B}$ can't just be any models and f can't just be any function with the correct domain and range to be useful in a swarmlet. There are assumptions implicit in a swarmlet like figure 5.5 that $\mathbb{A}$ and $\mathbb{B}$ are mutually consistent models of the same space. If $\mathbb{A}$ were a room-level model of the DOP center and $\mathbb{B}$ were an occupancy grid for a crater on Mars, there would not be much to be gained by attempting to match across their representations with an adapter. Similarly if $\mathbb{A}$ and $\mathbb{B}$ were consistent models of the same space, but f did not preserve the structure of the models, the matching result through f would be nonsense. Fortunately, we can use model embeddings to ensure $\mathbb{A}$, $\mathbb{B}$, and f make sense together. In this section we explore elementary embeddings for model substitution and the consequences of isomorphic embeddings for cross ontology inference.

Ontology Substitution with Elementary Embeddings

An elementary embedding of $\mathbb{A}$ into $\mathbb{B}$ is an assertion that formula over $\mathbf{A}$ yield the same result when evaluated in model $\mathbb{A}$ or model $\mathbb{B}$. If such an embedding is known to exist, only $\mathbb{A}$ or only $\mathbb{B}$ will suffice as a reference for evaluating semantic localization formula over $\mathbf{A}$. Therefore $\mathbb{A}$ (or $\mathbb{B}$) may be used in place of $\mathbb{B}$ (or $\mathbb{A}$) if the other ontology is unknown, prohibitively large to store, slow to access, or otherwise unavailable.

To make this concrete, we will consider the simple example of a temperature ontology expressed in units of Fahrenheit with an elementary embedding into another temperature

ontology in units of Celsius. These ontologies model a system in which 20 (an arbitrary number of) temperature sensors collect readings in an environment. The set of sensors S are labeled $\{s_1, s_2, ..., s_{20}\}$. The domain of $\mathbb{A}$ consists of the union of sensor labels and Fahrenheit temperatures,[5] as $\mathbf{A} = S \cup \mathbb{R}$. The domain of $\mathbb{B}$ is the same $\mathbf{B} = S \cup \mathbb{R}$ but with Celsius temperatures. Both models have a function $t : S \to \mathbb{R}$ which maps a temperature sensor to its sensed temperature. If for example t interpreted in ontology $\mathbb{A}$ has $t_{\mathbb{A}}(s_1) = 32$, t interpreted in $\mathbb{B}$ has $t_{\mathbb{B}}(s_1) = 0$ (with the subscript on t indicating interpretation). Both $\mathbb{A}$ and $\mathbb{B}$ have the standard functions (addition, subtraction, multiplication, absolute value, etc.) and ordering relation $\leq$ over $\mathbb{R}$. $\mathbb{A}$ and $\mathbb{B}$ are also equipped with constants signifying temperature differences in Celsius[6] (e.g. 5° is the constant representing a 5 degree C difference in temperature).

Given these definitions, its obvious $\mathbb{A}$ has an elementary embedding into $\mathbb{B}$, via $\mathbb{C}$ defined exactly as $\mathbb{A}$ and isomorphism $g : \mathbb{C} \to \mathbb{B}$ with $g(x)$ yielding $g(x) = x$ if x is a sensor and $g(x) = (x - 32) * (5/9)$ if $x \in \mathbb{R}$. In this case we can also define a function $f : \mathbb{A} \to \mathbb{B}$ which applies g directly from $\mathbb{A}$ to $\mathbb{B}$ since $\mathbf{A} = \mathbf{C}$, as $f(x) = (x - 32) * (5/9)$.

Now imagine we were interested in determining if two sensors had temperatures within 5 degrees C. We could write the sentence σ as $\sigma = \exists s_i, s_j \in \mathbf{A}\ |t(s_i) - t(s_j)| \leq 5^\circ$. Even if we would usually evaluate this sentence by testing $\mathbb{B} \models \sigma$, the elementary embedding ensures we will get the same result testing $\mathbb{A} \models \sigma$ instead. As desired we may substitute $\mathbb{A}$ for $\mathbb{B}$ when evaluating σ.

Cross Ontology Inference with Isomorphic Embeddings

The accessor pattern expressed in figure 5.5, is a solution to the problem where information (the location of the person for the robot to follow) is available in $\mathbb{A}$ but not $\mathbb{B}$ where it is needed. Use of adapters may be interpreted as a cross-ontology inference procedure wherein $\mathbb{B}$ gains information regarding the unknown location of the person on the occupancy grid from $\mathbb{A}$ and f. For example, the constant *person* is initially interpreted in $\mathbb{A}$ as $person_{\mathbb{A}} =$"$DOPCouch$" (indicating the location of that person at the "$DOPCouch$" label) and interpreted in $\mathbb{B}$ as $person_{\mathbb{B}} = \bot$. But if we knew $f($"$DOPCouch$"$) = (x, y)$, we could then infer infer $person_{\mathbb{B}} = f(person_{\mathbb{A}}) = (x, y)$. We may then construct a $\mathbb{B}'$ containing the inferred value for $person_{\mathbb{B}}$, satisfying $\mathbb{B} \sqsubseteq \mathbb{B}'$, and use $person'_{\mathbb{B}}$ to direct the robot.[7]

This example works because we can establish an isomorphic embedding between structures, however not from $\mathbb{A}$ to $\mathbb{B}$! It follows from definitions that $\mathbb{A}$ is not isomorphically embeddable into $\mathbb{B}$ by f: Any $\mathbb{C}$ satisfying $\mathbb{A} \subseteq \mathbb{C}$ must have $person_{\mathbb{C}} =$"$DOPCouch$".

[5] While physics tells us temperatures are limited to a particular range of values, e.g. -273 Celsius is absolute zero, we use the reals in this example for simplicity.

[6] While it is incongruous for this example that an otherwise all Fahrenheit ontology has constants representing temperature differences in Celsius, the definition of elementary submodel requires both models to have the same language, including constants. This example could be made to work with separate Fahrenheit and Celsius constants, but we would have to define a new variant of elementary submodel to include "language translation" of constants.

[7] Refer back to section 2.2 for definition 2.2.2 of $\sqsubseteq$ with respect to inference on open ontologies.

Applying $g : \mathbf{C} \to \mathbf{B}$ as a candidate isomorphismism results in $g(person_{\mathbb{C}}) = (x, y)$, but $person_{\mathbb{B}} = \perp$, violating $g(person_{\mathbb{C}}) = person_{\mathbb{B}}$. Instead we will show cross ontology inference for open ontologies $\mathbb{A} \sqsubseteq \mathbb{A}^*$ and $\mathbb{B} \sqsubseteq \mathbb{B}^*$ follows from establishing an isomorphic embedding of $\mathbb{A}^*$ into $\mathbb{B}^*$. The entire procedure may be summarized by the following idea: If something is known in $\mathbb{A}$ but isn't known in $\mathbb{B}$, use the matching f to bring it over to $\mathbb{B}'$. But if it was was already known in $\mathbb{B}$, don't change it in $\mathbb{B}'$.

Important note: In the definitions below where f is a function matching between ontologies we assume model domains are extended to include $\perp$, and that $f(\perp) = \perp$.

Definition 5.2.1 (Forward Cross Ontology Inference)**.** *Given idealized ontologies $\mathbb{A}^*$ and $\mathbb{B}^*$ and $f : \boldsymbol{A} \to \boldsymbol{B}$, for open ontologies $\mathbb{A} \sqsubseteq \mathbb{A}^*$ and $\mathbb{B} \sqsubseteq \mathbb{B}^*$, we may perform forward cross ontology inference to construct a $\mathbb{B}'$ via the following procedure:*

- For each constant k, if $k_{\mathbb{B}} = \perp$ and $k_{\mathbb{A}} = x \neq \perp$, then define $k_{\mathbb{B}'} = f(x)$. Otherwise define $k_{\mathbb{B}'} = k_{\mathbb{B}}$.
- For each n-place relation r, define $r_{\mathbb{B}'} = r_{\mathbb{B}} \cup \{(f(a_1), f(a_2), ..., f(a_n)) | (a_1, a_2, ..., a_n) \in r_{\mathbb{A}}\}$.
- For each m-place function h, if $h_{\mathbb{B}}(b_1, b_2, ..., b_m) = \perp$ and
  for some $(a_1, a_2, ..., a_3)$, $(f(a_1), f(a_2), ..., f(a_m)) = (b_1, b_2, ..., b_m)$, and
  $h_{\mathbb{A}}(a_1, a_2, ..., a_m) \neq \perp$, then define $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = f(h_{\mathbb{A}}(a_1, a_2, ..., a_m))$.
  Otherwise define $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = h_{\mathbb{B}}(b_1, b_2, ..., b_m)$.

Forward cross ontology inference is a procedure wherein information missing in one ontology can be brought in from another. Consider the temperature ontology example from above. Assume the Fahrenheit ontology $\mathbb{A}$ knows the temperature of every sensor with $\mathbb{A} = \mathbb{A}^*$. However the Celsius ontology $\mathbb{B}$ is missing the temperature of sensor 7, i.e. $t_{\mathbb{B}}(s_7) = \perp$. As an elementary embedding from $\mathbb{A}^*$ into $\mathbb{B}^*$ implies an isomorphic embedding we can use the same f (i.e. $f(x) = (x - 32) * (5/9)$) from before. We can use cross ontology inference to create a $\mathbb{B}'$ which knows the temperature of sensor 7. The inference case for functions where $t_{\mathbb{B}}(s_7) = \perp$ but $f(t_{\mathbb{B}}(s_7)) \neq \perp$ applies and we can set $t_{\mathbb{B}'}(s_7)$ accordingly.

Definition 5.2.2 (Reverse Cross Ontology Inference)**.** *Given idealized ontologies $\mathbb{A}^*$ and $\mathbb{B}^*$ and injection $f : \boldsymbol{A} \to \boldsymbol{B}$, for open ontologies $\mathbb{A} \sqsubseteq \mathbb{A}^*$ and $\mathbb{B} \sqsubseteq \mathbb{B}^*$, we may perform reverse cross ontology inference to construct an $\mathbb{A}'$ via the following procedure:*

- For each constant k, if $k_{\mathbb{A}} = \perp$ and for some x, $k_{\mathbb{B}} = f(x) \neq \perp$, then define $k_{\mathbb{A}'} = x$. Otherwise define $k_{\mathbb{A}'} = k_{\mathbb{A}}$.
- For each n-place relation r, define $r_{\mathbb{A}'} = r_{\mathbb{A}} \cup \{(a_1, a_2, ..., a_n) | (f(a_1), f(a_2), ..., f(a_n)) \in r_{\mathbb{B}}\}$.

- For each m-place function h, if $h_{\mathbb{A}}(a_1, a_2, ..., a_m) = \perp$ and for some x $h_{\mathbb{B}}(f(a_1), f(a_2), ..., f(a_m)) = f(x) \neq \perp$, then define $h_{\mathbb{A}'}(a_1, a_2, ..., a_m) = x$. Otherwise define $h_{\mathbb{A}'}(a_1, a_2, ..., a_m) = h_{\mathbb{A}}(a_1, a_2, ..., a_m)$.

Reverse cross ontology inference is a very similar procedure to forward cross ontology inference, only with the direction of inference reversed. In this case we bring information available in $\mathbb{B}$ into $\mathbb{A}$ instead of the reverse. Performing reverse cross ontology inference may require inverting f over its range in $\mathbf{B}$, or at least finding the preimage of f for some f(x)).

Again we consider the temperature ontologies as an example where $\mathbb{A}$ is ismorphically embedded into $\mathbb{B}$. This time consider the scenario where ontology $\mathbb{A}$ doesn't know the temperature of sensor 7 while $\mathbb{B}$ does. We can easily invert $f(x) = (x - 32) * (5/9)$ to $f^{-1}(x) = (x * 9/5) + 32$. The inference case for functions where $t_{\mathbb{A}}(s_7) = \perp$ and $t_{\mathbb{B}}(f(s_7)) = f(x) \neq \perp$ applies. We can apply f^{-1} to obtain the value of x from $f(x)$, and set $t_{\mathbb{A}}(s_7) = x$ with the temperature of sensor 7.

We next prove the correctness of forward cross ontology inference for isomorphically embedded models by proving $\mathbb{B} \sqsubseteq \mathbb{B}' \sqsubseteq \mathbb{B}^*$. It is important to show $\mathbb{B} \sqsubseteq \mathbb{B}'$ so that $\mathbb{B}'$ is consistent with $\mathbb{B}$ while potentially containing more information, and important to show $\mathbb{B}' \sqsubseteq \mathbb{B}^*$ so that $\mathbb{B}'$ does not disagree with $\mathbb{B}^*$.

Theorem 5.2.3. *Given idealized ontologies $\mathbb{A}^*$ and $\mathbb{B}^*$ with an isomorphic embedding from $\mathbb{A}^*$ into $\mathbb{B}^*$ (satisfying $\mathbb{A}^* \subseteq \mathbb{C}^*$ and $\mathbb{C}^* \cong \mathbb{B}^*$) with $g^* : \boldsymbol{C}^* \to \boldsymbol{B}^*$ as the isomorphism, let $f : \boldsymbol{A}^* \to \boldsymbol{B}^*$ be the restriction of g^* to $\boldsymbol{A}^*$, i.e. $f = g^*|_{\boldsymbol{A}^*}$. If $\mathbb{A} \sqsubseteq \mathbb{A}^*$ and $\mathbb{B} \sqsubseteq \mathbb{B}^*$, then forward cross ontology inference with $\mathbb{A}$, $\mathbb{B}$, and f will produce a $\mathbb{B}'$ satisfying $\mathbb{B} \sqsubseteq \mathbb{B}' \sqsubseteq \mathbb{B}^*$.*

Proof. We first show $\mathbb{B} \sqsubseteq \mathbb{B}'$, which is evident from definition 5.2.1 and definition 2.2.2 .

- For every constant k, there are two cases: either $k_{\mathbb{B}} \neq \perp$ and $k_{\mathbb{B}'} = k_{\mathbb{B}}$, or $k_{\mathbb{B}} = \perp$ and the value of $k_{\mathbb{B}'}$ does not matter. As in definition 2.2.2, if we were to define a function c mapping constant symbols in $\mathbb{B}$'s signature to $(\mathbf{B} \cup \{\perp\})$, $c_{\mathbb{B}} \leq c_{\mathbb{B}'}$
- Every relation $r_{\mathbb{B}'}$ is defined as the union of $r_{\mathbb{B}}$ and another set. Clearly $r_{\mathbb{B}} \subseteq r_{\mathbb{B}'}$
- For every m-place function h, there are two cases: either $h_{\mathbb{B}}(b_1, b_2, ..., b_m) \neq \perp$ in which case $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = h_{\mathbb{B}}(b_1, b_2, ..., b_m)$ or $h_{\mathbb{B}}(b_1, b_2, ..., b_m) = \perp$ in which case the value of $h_{\mathbb{B}'}(b_1, b_2, ..., b_m)$ does not matter. Therefore $h_{\mathbb{B}} \leq h_{\mathbb{B}'}$.

Next, we show $\mathbb{B}' \sqsubseteq \mathbb{B}^*$.

- Consider constant $k_{\mathbb{B}'}$. We show typical constant $k_{\mathbb{B}'}$ satisfies $c_{\mathbb{B}'} \leq c_{\mathbb{B}^*}$. If $k_{\mathbb{B}'}$ was assigned $k_{\mathbb{B}'} = k_{\mathbb{B}}$ it is consistent with $c_{\mathbb{B}'} \leq c_{\mathbb{B}^*}$ because we assumed $\mathbb{B} \sqsubseteq \mathbb{B}^*$. In the other case consider $k_{\mathbb{A}}$. We assumed $\mathbb{A} \sqsubseteq \mathbb{A}^*$ so there are two possibilities: Either $k_{\mathbb{A}} \neq k_{\mathbb{A}^*}$ (with $k_{\mathbb{A}} = \perp$) or $k_{\mathbb{A}} = k_{\mathbb{A}^*}$. The first possibility is consistent with $c_{\mathbb{B}'} \leq c_{\mathbb{B}^*}$, since $k_{\mathbb{B}'} = f(k_{\mathbb{A}}) = f(\perp) = \perp$. In the second possibility where $k_{\mathbb{A}} = k_{\mathbb{A}^*}$, we know $f(k_{\mathbb{A}^*}) = k_{\mathbb{B}^*}$ from the isomorphic embedding, leading to the likewise consistent assignment $k_{\mathbb{B}'} = k_{\mathbb{B}^*}$.

- Consider n-placed relation $r_{\mathbb{B}'}$. We show $r_{\mathbb{B}'} = r_{\mathbb{B}} \cup \{(f(a_1), f(a_2), ..., f(a_n))|(a_1, a_2, ..., a_n) \subseteq r_{\mathbb{A}}\} \subseteq r_{\mathbb{B}^*}$. By our assumption $\mathbb{B} \sqsubseteq \mathbb{B}^*$ we know $r_{\mathbb{B}} \subseteq r_{\mathbb{B}^*}$. Label the other part of the union which constitutes $r_{\mathbb{B}'}$: $y = \{(f(a_1), f(a_2), ..., f(a_n))|(a_1, a_2, ..., a_n) \in r_{\mathbb{A}}\}$. We must also show $y \subseteq r_{\mathbb{B}^*}$. The isomorphic embedding gives $y^* = \{(f(a_1), f(a_2), ..., f(a_n))|(a_1, a_2, ..., a_n) \in r_{\mathbb{A}^*}\} \subseteq r_{\mathbb{B}^*}$. As we assumed $\mathbb{A} \sqsubseteq \mathbb{A}^*$, we have $r_{\mathbb{A}} \subseteq r_{\mathbb{A}^*}$ implying $y \subseteq y^*$. With $y \subseteq y^*$ and $y^* \subseteq r_{\mathbb{B}^*}$, transitivity yields $y \subseteq r_{\mathbb{B}^*}$. Therefore $r_{\mathbb{B}'} \subseteq r_{\mathbb{B}^*}$.

- Consider m-placed function $h_{\mathbb{B}'}$. We show $h_{\mathbb{B}'} \leq h_{\mathbb{B}^*}$. If at $(b_1, b_2, ..., b_m)$ $h_{\mathbb{B}'}$ was assigned $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = h_{\mathbb{B}}(b_1, b_2, ..., b_m)$, it is consistent with $h_{\mathbb{B}'} \leq h_{\mathbb{B}^*}$ because we assumed $\mathbb{B} \sqsubseteq \mathbb{B}^*$. Consider the other case where $h_{\mathbb{B}'}$ was assigned $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = f(h_{\mathbb{A}}(a_1, a_2, ..., a_m))$. We assumed $\mathbb{A} \sqsubseteq \mathbb{A}^*$ so there are two possibilities: Either $h_{\mathbb{A}}(a_1, a_2, ..., a_m) \neq h_{\mathbb{A}^*}(a_1, a_2, ..., a_m)$ (with $h_{\mathbb{A}}(a_1, a_2, ..., a_m) = \bot$) or $h_{\mathbb{A}}(a_1, a_2, ..., a_m) = h_{\mathbb{A}^*}(a_1, a_2, ..., a_m)$. The first possibility is consistent with $h_{\mathbb{B}'} \leq h_{\mathbb{B}^*}$ because $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = f(h_{\mathbb{A}}(a_1, a_2, ..., a_m)) = f(\bot) = \bot$. In the second possibility where $h_{\mathbb{A}}(a_1, a_2, ..., a_m) = h_{\mathbb{A}^*}(a_1, a_2, ..., a_m)$, we know $f(h_{\mathbb{A}^*}(a_1, a_2, ..., a_m)) = h_{\mathbb{B}^*}(f(a_1), f(a_2), ..., f(a_m)) = h_{\mathbb{B}^*}(b_1, b_2, ..., b_m)$ from the isomorphic embedding, leading to the likewise consistent assignment $h_{\mathbb{B}'}(b_1, b_2, ..., b_m) = h_{\mathbb{B}^*}(b_1, b_2, ..., b_m)$.

□

The property $\mathbb{B} \sqsubseteq \mathbb{B}' \sqsubseteq \mathbb{B}^*$ (or $\mathbb{A} \sqsubseteq \mathbb{A}' \sqsubseteq \mathbb{A}^*$) justifies cross ontology inference. An inferred ontology will potentially have more information than it started with and still be consistent with the idealized ontology. Theorem 5.2.3 provides a justification for the intuition that ontologies must be modeling the "same thing" for their information to be combined. For most any real-world purpose, the programmer of an adapter is not going to sit down and write a formal proof of isomorphic embeddability regarding hypothetical idealized ontologies, but it is important to know such a relationship between spatial/contextual models does justify ontology matching as an inference procedure.

It can be seen the isomorphic embedding from $\mathbb{A}^*$ into $\mathbb{B}^*$ is sufficient but not necessary for correct forward cross ontology inference. Even if f is not an isomorphism at some $a \in \mathbf{A}$, forward cross ontology inference may still produce a $\mathbb{B}'$ satisfying $\mathbb{B} \sqsubseteq \mathbb{B}' \sqsubseteq \mathbb{B}^*$ as long as $\mathbb{B}$ already has a non-$\bot$ values wherever $f(a)$ would be assigned to a constant, relation, or function of $\mathbb{B}'$. For example, an extreme case of this occurs when $\mathbb{B} = \mathbb{B}^*$, resulting in $\mathbb{A}$ not playing any role in constructing $\mathbb{B}'$. Degenerate cases of inference aside, an isomorphic embedding is clearly an important property when matching between ontologies actually produces information for B'.

Provided f is invertible over its range in $\mathbb{B}$, a similar theorem to 5.2.3 can be proved for reverse cross ontology inference when an isomorphic embedding exists from $\mathbb{A}^*$ to $\mathbb{B}^*$ to show the inferred $\mathbb{A}'$ satisfies $\mathbb{A} \sqsubseteq \mathbb{A}' \sqsubseteq \mathbb{A}^*$. Since the isomorphism of an isomorphic embedding is defined as a bijection, the relationship between constants, relations, and functions of $\mathbb{A}$ and $\mathbb{B}$ goes both ways. We omit the proof to avoid repetition, but such a proof would be nearly identical to the proof of theorem 5.2.3 but with swapped roles of $\mathbb{A}$ and $\mathbb{B}$.

More repetitious definitions and theorems could be given by modifying cross ontology inference to work for an open matching function $\phi : \mathbf{A} \to \mathbf{B}$ where $\phi \leq f$ in the definition 2.2.1 sense of missing information. Such an alternate version of forward cross ontology inference would for instance only be able to bring information from $\mathbb{A}$ to $\mathbb{B}$ when $\phi(a) \neq \perp$. Such a ϕ with a partially known mapping between ontologies reflects the real-life circumstance in which the relationship between ontological domains is only partially known. We discuss active procedures for obtaining a ϕ in section 5.4.

5.3 Adapters as a Semantic Accessor Pattern

We now address the practical task of implementing a dynamic IoT service with discovered adapters. Recall that accessor discovery in the semantic accessor architecture (chapter 4 consists of querying a semantic repository for a contextually appropriate accessor. If the accessor is host compatible, it may be download and reified in a mutable accessor like those in figure 5.5. This suggests a similar approach for adapter discovery: if we can specify what it means for an adapter to be contextually relevant, we can test it for host compatibility, download, and reify it into a mutable to create a working swarmlet. In this section we propose a meta-ontology as the representation for the relationships between contextual ontologies. We give a working example of such a meta-ontology for the temperature example from section 5.2.

The Adapter Meta-Ontology

The elements of the adapter meta-ontology are other ontologies. As an ontology is formally a structure (as discussed in section 5.2), representing the full contents of multiple structures within the meta-ontology could easily get enormously complex and prohibitively large to represent. Fortunately a full representation is unnecessary as the adapter meta-ontology is primarily needed to determine which matchings between ontologies have been implemented by adapters. It is enough to consider an entire contextual ontology symbolically as a single concept within the adapter meta-ontology.

Recall the temperature ontologies example from section 5.2, but extended to include a Kelvin ontology in addition to the Fahrenheit and Celsius ontologies. A diagram illustrating the ontologies as circles and the matchings between them as arrows is given in figure 5.7. The arrow from F to C for example, signifies the ontology matching function from the Fahrenheit ontology to the Celsius ontology. Self-loops represent identity matchings that map every element of the ontology back to itself.

While this kind of representation is useful for visualizing the connections between ontologies, it isn't ideal as a meta-ontology for discovering adapters because it represents matchings as relations. An adapter is an accessor with more properties than just domain and codomain: it may require modules for host capabilities or remote services to compute its matching func-

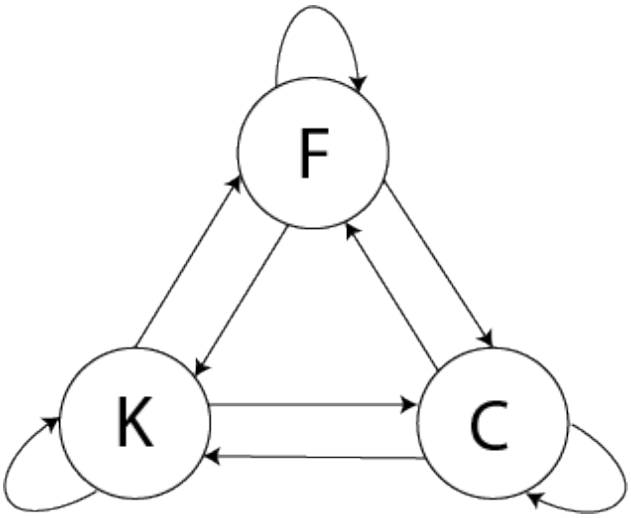

Figure 5.7: Diagram of Fahrenheit, Celsius, and Kelvin ontologies with arrows representing ontology matchings. Self-loops represent identity matchings.

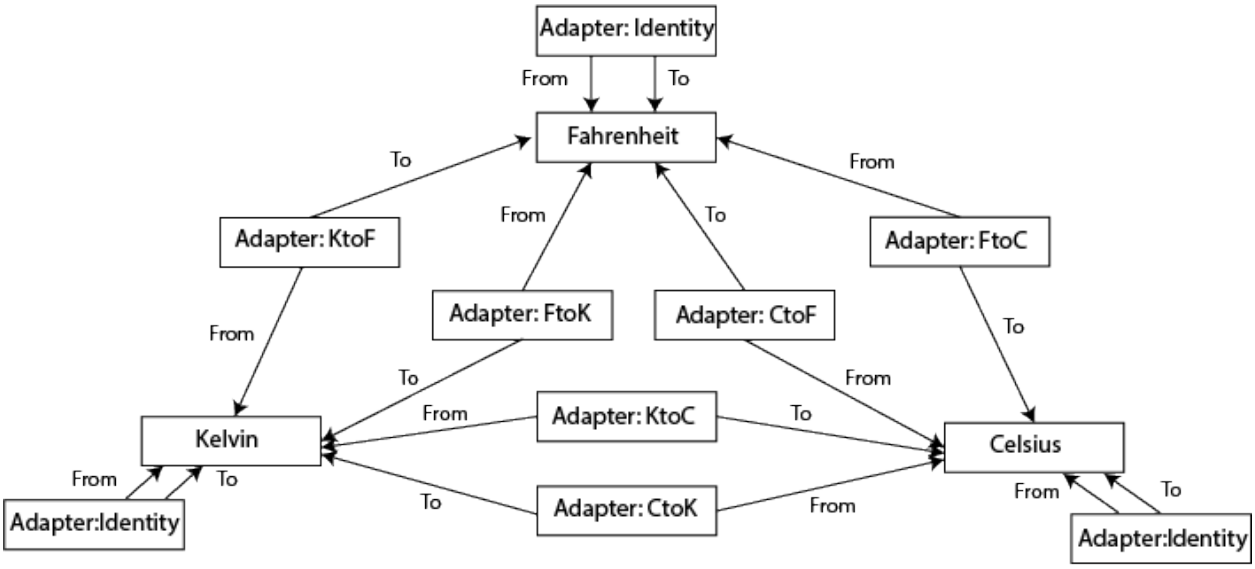

Figure 5.8: Diagram of Fahrenheit, Celsius, and Kelvin ontologies as concepts in a meta-ontology along with adapters implementing matchings *from* a domain ontology *to* a codomain ontology.

tion, extend other accessors, or in general be a part of the accessors ontology discussed in chapter 4.

Our proposed meta-ontology for this temperature example in figure 5.8 addresses this issue. The main change from figure 5.7 to figure 5.8 is the expansion of matching arrows into adapter objects, which may then have relationships to other concepts. Finding a useful

```
exports.setup = function(){
    this.extend('adapters/Adapter');
};

exports.matching = function(inValue){
    return inValue * (9.0/5.0) + 32.0;
};
```

Figure 5.9: JavaScript code listing for CelsiusToFahrenheit.js adapter

adapter from the meta-ontology is as simple as executing a SPARQL query on a semantic repository for an adapter with the correct "From" and "To" relations to the desired ontologies. Once the dynamic IoT swarmlet in figure 5.5 obtains the name of the ontology for sensor data and the name of the ontology for system data as the sensor and system accessors are discovered, the desired ontologies for matching are available for adapter discovery. If the sensor and system data already use the same ontology, an identity adapter is selected to complete the swarmlet.

Dynamic IoT Services with Adapters

Adapters are full-fledged accessors, and bring the same advantageous properties enjoyed by the component architecture discussed in chapter 3 to ontology matching in dynamic IoT services. Some matchings are complex (especially between spatial ontologies) and require extensive physical measurement to get right. Encapsulating the matching in a reusable accessor component means new systems can take advantage of the work done in creating the matching without duplicating the effort of building the matching again.

An interesting example of adapter reuse involves creating new matchings from transitive links in the adapter meta-ontology. If for example adapters were available for matchings from $\mathbb{X}$ to $\mathbb{Y}$ and from $\mathbb{Y}$ to $\mathbb{Z}$, but not from $\mathbb{X}$ to $\mathbb{Z}$, these existing adapters may be composed in series to create a previously undefined matching from $\mathbb{X}$ to $\mathbb{Z}$.

While an adapter may be an arbitrarily complex accessor, it may also be very simple. The complete implementation of the CelsiusToFahrenheit adapter is given in figure 5.9. CelsiusToFahrenheit extends the Adapter base class by overriding the default `exports.matching` function with a custom ontology matching. After testing the input against $\perp$ (to determine if the output should also be immediately set to $\perp$), the base class will invoke `exports.matching` on its input and output the result.

Using an adapter for cross-ontology inference can easily result in logical fallacies if appropriate care is not taken in addressing language inconsistencies. Ontologies may only by consistent for a portion of the language, so it may be necessary to explicitly define the language reduction and the resulting model reducts in the meta-ontology. As an example of the kind of accidental inference which may be made if inconsistent same-name relations are not

eliminated by a reduction, consider the following three statements (1,2, and 3) which are all independently reasonable but collectively imply a strange conclusion (4):

1. Love is in the heart.
2. I have a heart.
3. I am in the state of California.
4. Therefore, love is in the state of California.

We leave it to pop songs and philosophers to pass judgment on (4), but unintentionally combining a subjective statement of the human experience (1), a medical statement (2), and a geographic statement (3) could easily have an unexpected outcome in an automated reasoning system using transitivity for the "in" relation. To avoid such a circumstance we might augment the meta-ontology with language reduction information regarding a prohibition of the "in" relation after using an adapter to match from, for example, the medical ontology (2) to the geographic ontology (3).

5.4 Discussion and Future Work

Although we have presented theoretical and practical aspects of building dynamic IoT services with adapters, there are many future directions to go with this research. We overview some of these ideas in this section.

Probability and Confidence

Working with incomplete spatial/contextual information is a central element of this framework. We have already outlined how to make what is *unknown* explicit (the $\perp$ symbol), but we need to go further to express *uncertainty* when the content of a matching is known probabilistically. If some mapping relationships are more likely than others, the $\perp$ symbol has the unfortunate consequence of losing information.

The easiest (and least precise) way of dealing with uncertainty is to treat features of an ontology as expressing facts known up to a certain confidence threshold (eg. 95%). After eliminating all map features known at lower confidence (destroying potentially useful information), the remaining features may be treated as non-probabilistic facts with the caveat that in a large map or ontology some portion is likely to be wrong.

A more quantitative alternative is to modify the definition of the matching f from $f : \mathbf{A} \to \mathbf{B}$ to a probability measure $f : \mathbf{A} \to P(\mathbf{B} \times [0, 1])$, where P is the powerset operator and $\forall a \in A; M(a) = \{(b_0, p_0), (b_1, p_1), ...\}$ the p_i represent probabilities that integrate or sum to one over $M(a)$. This method represents uncertainty about location for each element of the map domain, but it does not express potential correlations in the random variables we are attempting to describe with $f(a)$ and $f(a')$.

Instead of associating a distribution with $f(a)$, another alternative is to associate probability with the matching f itself. We might define a probability measure F as a family of matchings $f_0, f_1, f_2, ...$ with each $f_i : A \to B$ associated with a probability that sums or integrates to one over F. This latter approach is more general, because it can represent correlation. For example, if it is known $f(a)$, $f(a')$, and $f(a'')$ have a particular triangular arrangement in $\mathbf{B} = \mathbb{R}^2$, but the location of the triangle is unknown (perhaps either centered at $(0, 0)$ or at $(1, 1)$) only this second probabilistic method can unambiguously represent a distribution over triangular arrangements.

In a distributed system where maps are obtained by distinct entities, reconciling competing perspectives on the world is an additional complication. The Dempster-Shafer theory of evidence [85] is a generalization of the Bayesian theory of subjective probability judgments, and a possible avenue of research for accessing how far fallible systems should be trusted. Even trustworthy systems have a blind spot with respect to the passage of time. When tracking a mobile target, a sufficiently long duration ought to correspond to decreased confidence in the accuracy of map data. One simple possibility is to discount information, providing an exponential decay of confidence over time.

Connection with Milner's bigraphs

The combination of a path map and a containment map (specifically a set of trees) over the same objects is a special mathematical entity known as a bigraph [68]. Bigraphs were introduced by Milner to model computation embedded at a particular location in space within a communication network. Reactions and transitions can be defined for a bigraph to describe evolution of the system over time, facilitating verification for ubiquitous computing systems.

Pereira's dissertation [74] introduces the BigActor model, which augments bigraphs to include Actors hosted at bigraph nodes that know the name of the computing machine on which they are executing. Actors can start other actors, send messages, learn about the structure of the bigraph in which they exist, change the bigraph, and migrate to another host. Pereira uses BigActors for spatial programming in a case study where UAVs are sent to monitor an oil spill in the ocean.

We foresee applications where location models might be augmented with a special bigraphical analysis component to reason about the evolution of spatial ontologies over time for verification purposes or support a BigActor spatial programming system. The methods in this chapter could facilitate the fusion of metric information with a bigraph's containment and topological knowledge.

Active Ontology Matching

We have not yet thoroughly discussed the process by which an adapter implementing an ontology matching is created. The adapter for the robot following assistant demo was constructed by a human manually specifying a correspondence from room labels to occupancy

grid coordinates. The temperature adapters implement known functions between these standard units. These two approaches (i.e. manual human configuration and a priori human knowledge) are valid ways of buildings adapters, but an exciting future possibility involves finding matchings automatically via machine learning and ontology matching algorithms from the literature.

We believe active ontology matching is a particularly compelling avenue for this research. **Active ontology matching** entails *initiating real-world actuation and sensing of devices in order to collect information useful in establishing an ontology matching.* The previous approaches we have discussed are passive in the sense that they involve no actions being taken to collect new information. An active ontology matching procedure for the robot following assistant might for example introspectively use the FIND location sensor to track the Turtlebot (instead of the human) as it autonomously navigates around the space. When the FIND system reports a room label for the robot at the same time it is at a particular occupancy grid location, evidence has been collected to support a matching from the observed FIND room label and the occupancy grid coordinates. A similar active procedure could be used to find the Turtlebot in a video-feed: trigger a Turtlebot movement when the rest of the video frame is still. The hope is these active measures would further automate the process of running a dynamic IoT service.

Improved accuracy within contextual models is a interesting side benefit of automatically combining spatial/contextual models. For example, many localization systems have seen great success by employing sensor fusion techniques to combine data from sensors with diverse failure modes. Adapters could be used to automate this procedure.

Privacy

The quality of information in spatial/contextual ontologies offers an intrinsic trade-off between privacy and usefulness. If the system knows too little about a user's location, the quality of its services are diminished, but if it knows too much about the user's location, privacy is lost. The ontology matching formalism could help users navigate the grey zone between the two extremes by determining which questions about location may be answered without giving away full position.

A key advantage of representing position with a collection of interconnected spatial ontologies is the ability to limit access control for a very specific aspect of a map to a desired user-group. The conventionally considered endgame of localization today of 3D coordinates for everything on the same map does not facilitate degrees of access to information because all entities are available together.

Conclusion

We believe dynamic IoT services have tremendous potential for the future of the IoT. In this chapter we address the problem of semantic interpretation across independently engineered components with adapters. An adapter is an accessor component which implements an

ontology matching across the spatial/contextual models in which independently engineered sensors and systems were designed. Adapters fit into a swarmlet design pattern for dynamic IoT services. We give the robot following assistant demo as a proof of concept implementation of this pattern, integrating a sensor and system with two distinct location models.

Returning to the formulation of semantic localization for spatial/contextual ontologies from chapter 2, we defined forward and reverse cross ontology inference procedures and proved their correctness given certain model theoretic assumptions. We showed how different assumptions justify ontology substitution. Finally, we gave a semantic accessor pattern for adapters leveraging the approach for accessor discovery from chapter 4 with the adapter meta-ontology.

As spatial/contextual models become increasingly valuable to ubiquitous devices, malicious adversaries have further motivations to corrupt the data. Ontology matching from abstract relational ontologies to Euclidean representations facilitates consistency checking between reported positions to catch liars. For example, Yang et al. show how in a sensor network graph of distance measurements, rigidity matroid theory can be used to isolate impossible edges [104]. We take a similar approach with a new algorithm in chapter 6 by interpreting IoT devices as sensor network nodes to identify incorrect distance measurements between IoT devices.

Chapter 6

Gordian: Formal Reasoning Based Outlier Detection for Secure Localization

Accurate spatial models are already crucial for CPS control and location-based applications, and stand to become even more important as systems with dynamic context-aware functionality become more commonplace. In the previous chapters we have discussed spatial/contextual models and platforms for IoT composition. However, many applications that depend on location data are subject to dangerous failure-modes when inconsistent spatial ontologies are combined or the CPS providing location information fails. For example in the hospital tracking context, a localization system failure could be life threatening. Likewise, a platoon of military vehicles trying to localize themselves with respect to each other while some of the cars are being hijacked cannot afford location errors. Due to demonstrated vulnerabilities in Global Navigation Satellite System (GNSS) spoofing [39], an adversary may likewise mislead or attempt to crash an autonomous vehicle or robot.

Fortunately, many localization systems with sensors embedded in the environment resemble distributed sensor networks (eg. [51]) where a measurement failure at a particular sensor may be isolated from other sensors. This distributed sensing can provide a great deal of redundancy in position estimation to aid in detecting and eliminating location-attacks and coordinated sensor failures. We explore this scenario in this chapter through the lens of range-based localization: distance measurements are known between sensors, only some of which have known locations, and we seek to localize the entire network in the presence of adversarially corrupted measurements.

We propose GORDIAN[1], an attack and coordinated sensor failure detection algorithm. Detecting and mitigating attacks on sensor data is, in general, a combinatorial problem [73], which has been typically addressed by either brute force search, suffering from scalability issues [73], or via convex relaxations using algorithms that can terminate in polynomial time [24] but are not necessarily sound. On the other side, recent advances in combinatorial

[1]The name GORDIAN is a reference to a legend of Alexander the Great in which he "untied" the impossible Gordian Knot by slicing it in half with his sword.

search techniques and in particular those used in Satisfiability Modulo Theories (SMT) solvers showed combinatorial problems can be cast into smaller problems that can be solved efficiently. As we demonstrate in this chapter, these new Satisfiability Modulo Convex (SMC) solving techniques [87] are a neat fit for range-based attack detection.

This chapter presents the following contributions:

- Sufficient topological and combinatorial conditions for attack detection in a noiseless network.
- GORDIAN, the first provably sound and complete coordinated attack detection algorithm over well-formed noiseless networks.
- A novel trilateration counterexample generation procedure for GORDIAN's SMT solving architecture that makes combinatorially intractable attack detection problems practically solvable in reasonable time.
- A localization algorithm appropriate for noise on both distance measurements *and* anchor coordinates.
- A generalization of the standard graph-embeddability problems studied in noiseless localization to a notion of *approximate embeddability*, appropriate for noise.
- A convex decision procedure for testing approximate embeddability of noisy networks at a desired confidence level, enabling the extension of GORDIAN to the noisy case.

In Section 6.1 we introduce conventional and secure localization algorithms, define our formal model of localization problems and introduce rigidity theory. Next in Section 6.2 we define our threat model, the attack detection problem, and identify an attack tolerance property for which we prove sufficient topological and combinatorial conditions. We present GORDIAN's architecture in Section 6.3, prove the algorithm sound and complete, and elucidate the algorithms for embeddability testing and counterexample generation. Section 6.4 extends localization, embeddability testing, and GORDIAN to the noisy case, which we evaluate in Section 6.5. We conclude in Section 6.6.

6.1 Background and Related Work

Researchers have proposed a wide variety of localization schemes [62] including techniques as diverse as RF signal strength and fingerprinting [82, 38], propagation time of an ultrasonic pulse [78, 51], and range-free techniques[36]. Many of these techniques assume complete trust of the entire localization system. The field of secure localization goes a step further to explore methods that work in the presence of malicious attacks or failures.

Localization Networks

We apply a simple and common model [23, 69, 90, 12, 61, 104, 82, 4] of range-based localization problems: Incomplete pairwise distance measurements are known between devices. Some devices, referred to as anchors, have known positions and other unlocalized devices do not.

More formally, we assume a set of n nodes representing sensors $\mathcal{S} = \{s_1, s_2, \ldots, s_n\}$ with $S = \{1, 2, ..., n\}$ being their index set. $S_a \subseteq S$ represents anchor nodes, defined as nodes with *a priori* measured locations $p_a : S_a \to \mathbb{R}^2$. The rest, $S_x = S \setminus S_a$, have unspecified locations. An undirected graph $G = (S, E)$ is a natural model for the topology of a sensor network where the sensor nodes are treated as graph nodes and edges E represent measured internode distances. Let $E_a \subseteq E$ represent the edges (i, j) such that both $i, j \in S_a$ and $E_x = E \setminus E_a$. The weighted extension of G is given as $G_d = (S, E, W)$ where $W : E \to \mathbb{R}^+$. Combining all of the above, we formally define:

Definition 6.1.1 (Localization network)**.** *A localization network N is a tuple (S, E, W, p_a) such that S is a set of nodes, E is a set of edges, $W : E \to \mathbb{R}^+$, $p_a : S_a \to \mathbb{R}^2$, where $S_a \subseteq S$.*

Let the function $p : \mathcal{S} \to \mathbb{R}^2$, be a "placement", assigning coordinates $p(s_i) \in \mathbb{R}^2$ to each sensor node. Now assume p^* is a placement representing ground truth positions of the nodes. The euclidean distance from node s_i to s_j is given by $d_{ij} = \|p^*(s_i) - p^*(s_j)\|_2$. With $\restriction$ as the set restriction operator, we are now equipped to pose the *exact* (noiseless) *localization* problem as follows:

Problem 6.1.2 (Exact localization)**.** *Given a localization network $N = (S, E, W, p_a)$ with an unknown ground truth placement p^* s.t. $W(e_{ij}) = d_{ij}$ and $p_a = p^* \restriction_{S_a}$, find p^*.*

Figure 6.1a shows an example localization network and placement which will be a long-running example in this chapter. It has six anchors (nodes $1 - 6$) represented by squares and one non-anchor (node 7) represented by a circle. Lines in the diagram represent measured internode distances, where the length of a measurement (i.e. $W(e_{xy})$) from s_x to s_y corresponds to the length of the line from node x to node y. The placement of the nodes is intended to correspond to their position on the page. In other words this is an embedded graph, not an abstract graph representation. Lines from anchors to anchors are feinter than the lines to node 7 because we wish to draw attention to the latter. Figure 6.1 (a) is an example of an embedding (also called a realization) because all the measurements are consistent with the placement. Were figure 6.1 (a) to be an inconsistent placement, one or more of the lines in the diagram would fail to meet up with a node at its endpoints. This is the case in figure 6.3.

Rigidity

Perhaps the most important question to ask about a localization network in the context of attack detection is whether or not p^* is the unique consistent placement with respect to N.

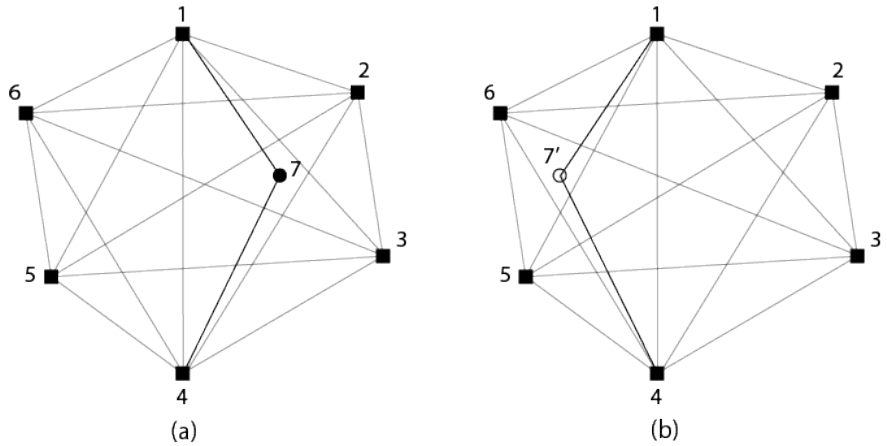

Figure 6.1: Two different embeddings (a) and (b), for the same localization network that is rigid but not GGR.

This property is known as *unique localizability*. If a localization network has two distinct embeddings in the absence of attack, an adversary does not have to take any action to create ambiguity in the localization result as is the case for the two embeddings shown in figure 6.1. The hollow circle in figure 6.1b for node $7'$ represents the alternative consistent placement for $p(7) \neq p^*(7)$. Even without attacks, there are two ambiguous localization solutions.

Eren et al. [23] identified Global Generic Rigidity (GGR) as the link between sensor network localization, the mathematical concept of a framework, and the theory of structures from mechanical, civil engineering, and physics. Intuitively, a framework is a collection of solid rods connected at flexible joints. A rigid framework will not deform when perturbed in a planar direction. A GGR framework is not only immune to planar deformation, it cannot be discontinuously manipulated (such as the 3-dimensional flip of node 7 along the line connecting nodes 1 and 4 in figure 6.1) into another configuration. Eren et al. also proved a network N is uniquely localizable if and only if its corresponding grounded graph (the framework obtained by treating a localization network's graph of measurements, G, as rods) is GGR. For simplicity we will write that N is GGR when its grounded graph is GGR. [23]

A globally rigid framework can be identified by topological conditions: G is *3-connected* (at least three edges must be removed to partition G) and G is *redundantly rigid* (any one edge may be removed from G and the resulting G' is still rigid) [40]. We further assume that the frameworks under discussion are generic, i.e. the coordinates of p are algebraically independent over the rationals. Essentially, the generic requirement forces the framework's p to not be entirely co-linear or otherwise degenerate. The GGR property is efficiently testable by a randomized algorithm [32]. We refer the interested reader to [23, 4] for a more thorough presentation on rigidity and localization.

Localization Algorithms

Localization algorithms attempt to solve the exact localization Problem 6.1.2. From a complexity perspective, just determining if a graph has an embedding (that preserves the d_{ij}) is known to be NP hard [83]. Researchers tackle intractability through two broad classes of methods: local methods that enable each node to determine its location from its neighbors without seeing the big picture and global methods that simultaneously localize all nodes from a perspective external to the network. GORDIAN makes use of both.

The most basic local algorithm, provided by Eren et al. is *iterative trilateration*, but it is only possible for a specific kind of *trilateration graph*. The definition of a trilateration graph is given in [23], but informally it can be thought of as a localization network that admits the following localization procedure: Begin with a set of three nodes $\{s_i, s_j, s_k\}$ with known locations (initially these can be anchor points). Find a node s_n that is currently without known location and is connected to three nodes s_i, s_j, and s_k in the known set. Draw three circles centered $p(i)$, $p(j)$, and $p(k)$ with radius d_{in}, d_{jn}, and d_{kn} respectively. The value of $p(n)$ is given by the unique intersection of the circles. Add s_n to the known set, and repeat the above procedure until all nodes are in the known set.

Eren et al. prove trilateration graphs are uniquely localizable (an important property for GORDIAN's counterexamples phase – see algorithm 2), and can be localized in polynomial time. However the procedure is incomplete in the sense that it fails to localize classes of uniquely localizable graphs such as bipartite and wheel graph networks [69].

Alternatives to the node-centric localization methods discussed above use some sort of optimization framework to localize all nodes at once [90, 12, 13, 4]. Although actual formulations vary, these approaches frame localization as an optimization problem and (very broadly speaking) aim to minimize the sum of some sort of squared errors resembling

$$\underset{p(i), i \in S_x}{\operatorname{argmin}} \sum_{(i,j) \in E_x} \frac{|\|(p(i) - p(j)\|_2^2 - W(e_{ij})^2|}{W(e_{ij})^2}$$

$$s.t. \quad \forall i \in S_a \; p(i) = p_a(i) \tag{6.1.1}$$

where $p(i) \in \mathbb{R}^2$ is a decision variable representing the estimated location of sensor i, and $W(e_{ij})$ is the measured distance between i and j. Since distances are squared, the denominator $W(e_{ij})^2$ normalizes the influence of large edges in optimization.

These methods commonly rely on a relaxation of the general optimization problem in Equation 6.1.1 from a non-convex program in two dimensions to a Semidefinite Program (SDP) in a higher dimensional space (refer to (6.3.3) for the statement of the Biswas-Ye SDP relaxation (BY-SDP) used by GORDIAN). The relaxation was first proposed by Biswas and Ye [12] with good empirical performance, then proved to have important theoretical connections to rigidity and unique localizability [90]. Most significantly, it is shown [90, Theorem 4.2] that a two-dimension graph is uniquely localizable if and only if the max-rank BY-SDP solution is rank 2.

In this chapter, we seek to address the *secure localization* problem with our GORDIAN algorithm. According to Zeng et al.'s survey [106], secure localization methods in the literature fall into three broad categories: *prevention methods* which prevent the sensor network from collecting bad data in the first place, *detection methods* which identify and remove bad localization data, and *filtering methods* which are robust to bad localization as part of the localization procedure. Under this taxonomy, GORDIAN is a centralized detection method designed to identify and eliminate distance-measurement attacks (i.e. attacks that corrupt inter-node distance measurements) and anchor position change attacks before localization.

6.2 Localization Attack Detection

As described in Sec. 6.1, even in the absence of attacks there are certain graph properties that must be present in the localization network for the localization problem to be well-posed. It should be no surprise then that stronger properties are required for the attack detection problem to be well-posed in the presence of attacks. Although conditions for the number of required non-malicious anchors are presented in [107], and rigidity conditions for outlier detection are addressed in [104, 105] this chapter is the first of our knowledge to give conditions and a systematic procedure for attack detection in the presence of adversarially *coordinated* sensor failures.

Threat Model

We consider two agents: a *localization system* that attempts to solve Problem 6.1.2 and an *adversary* that interferes by corrupting some of the measurements seen by the localization system with the goal of causing the system to incorrectly localize one or more sensors to the wrong locations. Thus, the adversary has potentially tampered with the localization system's ranging measurements[2] $m_{ij} = W(e_{ij}) * (1 + \delta_{ij})$ and anchor positions $a_i = p_a(i) + \alpha_i$. The $\delta_{ij} \in \mathbb{R}$ and $\alpha_i \in \mathbb{R}^2$ values are controlled by the adversary. For now we assume a noiseless scenario: if an edge measurement m_{ij} (or an anchor position a_i) is attack-free or "clean", then $m_{ij} = d_{ij}$ ($a_i = p^*(i)$). Furthermore we assume from the system's perspective there is no otherwise distinguishing feature between clean and corrupted edges. We formalize an attack as an *attack profile* $\hat{\eta} = (b, c, m, a)$ where b is the set of attacked edges, c the set of attacked anchors, m the new weights on the attacked edges, and a the new locations of attacked anchors.

A localization network under attack profile $\hat{\eta}$ is modified with these observed values in the expected way: the graph G_d is weighted by m_{ij} instead of d_{ij} and anchor positions are given by $p_a(i) = a_i$ instead of $p_a(i) = p^*(i)$. The localization network N under attack $\hat{\eta}$ will be denoted $\hat{\eta}(N)$, where $\hat{\eta}(\cdot)$ is the *application* of an attack profile to a localization network. For example, in figure 6.2, the localization network N depicted in (a) has the

[2]Ranging measurements are corrupted multiplicatively (not additively) to avoid cross terms between distance and error when squaring $W(e_{ij})^2$ in the noisy case (Section 6.4).

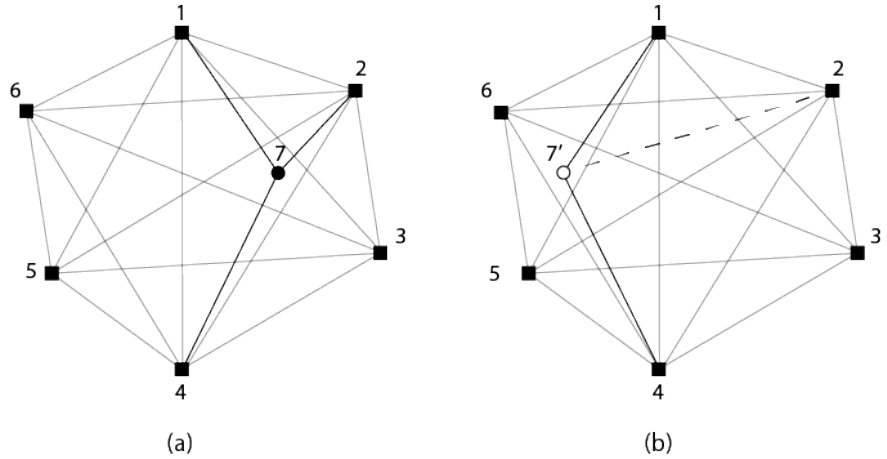

Figure 6.2: An illustration of unique embeddings for a GGR localization network before attack (a) and after (b). The attack on edge (2,7), represented by the dashed line, still yields a consistent embedding in (b). This localization network is susceptible to undetectable attacks.

correct distance between nodes 2 and 7: $\|p(2)-p(7)\|_2^2$. But (b) depicts an embedding of the attacked localization result $\hat{\eta}(N)$, where attack profile $\hat{\eta} = (\{(2,7)\}, \emptyset, \{\|p(2) - p(7')\|_2^2\}, \emptyset)$.

Problem Formulation

In this section we introduce formal notation for discussing attacks in a localization network. If the system can identify the attack and remove all corrupted data from the localization network it observes, the system is free to solve Problem 6.1.2 using an ordinary, insecure, localization algorithm. We name this prior task the *attack detection* problem.

Given the graph (S, E) of a localization network, we define an *attack hypothesis* $\eta = (b, c)$ where $b \subseteq E, c \subseteq S_a$. Intuitively, this is a guess as to which anchors and edges are under attack. Clearly, for each true attack, an attack profile $\hat{\eta} = (b, c, m, a)$ induces an attack hypothesis $\eta = (b, c)$ by simply "remembering" the identities of attacked edges/anchors and "forgetting" the values of the attack. A natural partial order is defined on attack hypotheses, where $(b, c) \preceq (b', c')$ iff $b \subseteq b'$ and $c \subseteq c'$ (as set inclusion). We will say that $\hat{\eta}$ (or η) is an $(\overline{s}, \overline{t})$*-attack profile (hypothesis)* if $|b| \leq \overline{s}, |c| \leq t$. The space of all $(\overline{s}, \overline{t})$-attack profiles (hypotheses) for a localization network N will be denoted $\hat{H}_N(\overline{s}, \overline{t})$ $(H_N(\overline{s}, \overline{t}))$, or, if N is clear from the context, just $\hat{H}(\overline{s}, \overline{t})$ $(H(\overline{s}, \overline{t}))$. If $\hat{\eta}(N) = N$, we say that $\hat{\eta}$ is *trivial* with respect to N.

Given a localization network N and an attack hypothesis η, we denote $N \setminus \eta$ the localization network with the hypothesized attacked edges and anchors removed. Since an attack profile $\hat{\eta}$ induces an attack hypothesis η, we will sometimes abuse notation and write $N \setminus \hat{\eta}$ for an attack profile $\hat{\eta}$, which means we remove from N the attack hypothesis induced by $\hat{\eta}$. We pose the attack detection problem for secure localization as follows.

Problem 6.2.1 (Noiseless Attack Detection)**.** *For a localization network N and an attack profile $\hat{\eta}$, given only $\hat{\eta}(N)$, find a hypothesis $\zeta \in H(\overline{s}, \overline{t})$ such that $\eta \preceq \zeta$ (where η is the hypothesis induced by $\hat{\eta}$).*

The crux of successful attack detection is to make the attacks stand out in some way from the clean measurements of the localization network. If corrupted edges and anchors make up only a small part of a localization network they can be identified as the minimal part of the network that is incompatible with the rest. This forms the general idea behind voting-based attack detection schemes, e.g., [61]. Drawing inspiration from Yang et al.'s rigidity-based outlier detection technique [104], we use the *embeddability* of a localization network, to define an $(\overline{s}, \overline{t})$-attack tolerant (abbreviated as $(\overline{s}, \overline{t})$-AT) localization network that admits incompatibility-based attack identification for up to $\overline{s}$ distance measurement attacks and $\overline{t}$ anchor position attacks.

Unlike other secure localization algorithms, our approach seeks to provably identify *all* attacks in networks meeting requirements we will lay out in theorem 6.2.6. As such, $\overline{s}$ and $\overline{t}$ are assumptions which must be made in advance regarding the maximum number of attacks to search for in the network. To see why it is necessary to make such assumptions about the network, consider how attack detection is impossible if $\overline{s}$ and $\overline{t}$ are the size of the network. Consistency-based outlier detection would be impossible because an attack detection algorithm could simply return the entire network as the attack, and nothing would actually be detected.

Instead $\overline{s}$ and $\overline{t}$ are best interpreted as parameters representing assumptions made by an attack detection algorithm. If $\overline{s}$ and $\overline{t}$ are too small, an attack detection algorithm may miss attacks. If $\overline{s}$ and $\overline{t}$ are too large, the algorithm may mistakenly identify clean measurements while ignoring the real attack. Theorem 6.2.6 gives guidance on how large $\overline{s}$ and $\overline{t}$ may be in a given network for guaranteed attack detection success. To clarify what we mean when we talk about part of a localization network, we introduce sub-localization networks in definition 6.2.2.

Definition 6.2.2 (Sub-localization Network)**.** *A localization network $N' = (S, E', W', p'_a)$ is a sub-localization network of $N = (S, E, W, p_a)$ (denoted $N' \subseteq N$) when $E' \subseteq E$, $W' = W \restriction E'$, and $p'_a = p_a \restriction S'_a$. If in addition, $|E \setminus E'| \leq \overline{s}$ and $|S_a \setminus S'_a| \leq \overline{t}$, we write $N' \subseteq_{\overline{s},\overline{t}} N$.*

Given an attack profile (hypothesis) $\hat{\eta}$ on N, we define its restriction $\hat{\eta} \restriction_{N'}$ by simply removing any attacks on edges or anchors that are not in N'. By an abuse of notation, we will sometimes write $\hat{\eta}(N')$ instead of $\hat{\eta} \restriction_{N'} (N')$. A sub-localization network $N' \subseteq N$ also induces an attack hypothesis which we denote $\eta_{N'}$, such that $N' = N \setminus \eta_{N'}$. We can now define:

Definition 6.2.3 ($(\overline{s}, \overline{t})$-Attack Tolerance)**.** *A localization network N is $(\overline{s}, \overline{t})$-attack tolerant if $\forall \hat{\eta} \in \hat{H}(\overline{s}, \overline{t})$ and $\forall \eta \in H(\overline{s}, \overline{t})$, for $N' = N \setminus \eta$ (in which case $N' \subseteq_{(\overline{s},\overline{t})} N$), if $\hat{\eta}(N')$ is consistent then $\hat{\eta}$ is trivial with respect to N', i.e. $\hat{\eta}(N') = N'$.*

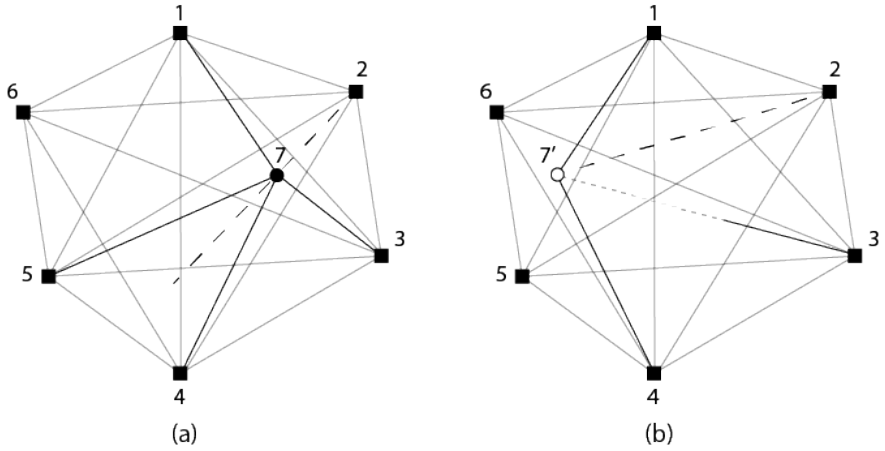

Figure 6.3: Adding a redundant edge (3,7) to the example in figure 6.2 (b) satisfies Lemma 6.2.4. Neither the placement in (a) nor the placement in (b) is an embedding: (a) is incompatible with edge (2,7) and (b) is incompatible with (3,7)

Intuitively, the definition says that if we guess an incorrect $(\overline{s}, \overline{t})$ attack hypothesis and did not remove *all* of the attacked edges and anchors, the application of the real attack $\hat{\eta}$ to the resulting sub-localization network will necessarily leave it inconsistent. Therefore, attacks in an $(\overline{s}, \overline{t})$-AT localization network can be identified by consistency checking. Clearly, GGR is an insufficient criterion, as shown in figure 6.2b where the attacked localization network is still embeddable; attack tolerance requires something more.

Conditions for Attack Tolerance

We begin by first considering what it takes for a localization network to be $\overline{s}$-AT in the absence of anchors (and anchor attacks). In our analysis a $\overline{s}$-redundantly generically globally rigid (abbreviated as $\overline{s}$-GGR) network, is a localization network that remains GGR after the removal of up to any $\overline{s}$ edges from its graph. We start with the following lemma about $\overline{s}$-GGR localization networks, illustrated by figure 6.3:

Lemma 6.2.4. *Let N be a consistent $\overline{s}$-GGR localization network, $\hat{\eta} \in \hat{H}(\overline{s}, 0)$. $\hat{\eta}(N)$ is conistent if and only if $\hat{\eta}$ is trivial.*

Proof. Clearly, if $\hat{\eta}$ is trivial, $\hat{\eta}(N) = N$ is consistent. On the other hand, suppose $\hat{\eta}(N)$ is consistent. $\hat{\eta}(N) \setminus \hat{\eta} = N \setminus \hat{\eta}$ is also consistent (removing edges cannot make it any less consistent), and is GGR, since we removed no more than $\overline{s}$ edges from N. It therefore has exactly one realization (up to isometry), which induces the same edge weights on both N and $\hat{\eta}(N)$ □

Theorem 6.2.5. *If N is $2\overline{s}$-GGR, then N is $(\overline{s}, 0)$-AT.*

Proof. Let $N' \subseteq_{(\overline{s},0)} N$ and $\hat{\eta}$ an $(\overline{s},0)$-attack profile on N such that $\hat{\eta}(N')$ is consistent. N' is a consistent $\overline{s}$-GGR localization network, and therefore $\hat{\eta}(N') = N'$ by Lemma 6.2.4. □

The above results do not consider anchor attacks, but more importantly, they do not rely on the placement (or existence) of anchors at all, and are true up to isometry. We can therefore state the following.

Theorem 6.2.6. *If N is $2\overline{s}$-GGR and $|S_a| \geq 2\overline{t}+3$, then N is $(\overline{s},\overline{t})$-AT.*

Proof. We consider an attack $\hat{\eta} \in H(\overline{s},\overline{t})$ and $N' \subseteq_{(s,t)} N$ such that $\hat{\eta}(N')$ is consistent. We must show the attack is trivial with respect to N'.

Clearly, since N' is $\overline{s}$-GGR the attack cannot contain any edges in N', or by Lemma 6.2.4 $\hat{\eta}(N')$ would be inconsistent. But without edge attacks, $\hat{\eta}(N')$ has exactly one realization up to isometry (it is $\overline{s}$-GGR, and in particular, GGR), and at least 3 unattacked anchors. These anchors uniquely localize the network, and so any anchor attack would make $\hat{\eta}(N')$ inconsistent. The attack is therefore trivial. □

Theorem 6.2.6 gives sufficient conditions for attack tolerance. A full characterization of the necessary and sufficient conditions for $(\overline{s},\overline{t})$-AT attack detection are beyond the scope of this chapter. The challenge of this characterization lies in evaluating the attack tolerance of a localization network that is not sufficiently redundantly GGR, but makes up the difference with anchor information. For an extreme example consider a sparsely connected localization network with no anchor attacks where every node is an anchor. Edge attacks are detectable but not by rigidity.

Algorithm 1 Attack Detection

procedure AttackDetection(`localization network` $\hat{\eta}(N)$, $\overline{s}$, $\overline{t}$)
  for $e_{ij} \in E$ and $a_k \in S_a$ **do**
    Declare pseudoboolean indicator variables b_{ij} and c_k ▷ 1 represents corrupted, 0 represents clean
  $C \leftarrow (\sum_{(i,j)\in E} b_{ij} \leq \overline{s}) \wedge (\sum_{k\in S_a} c_k \leq \overline{t})$ ▷ C is a set of pseudoboolean SAT clauses
  while Satisfiable(C) **do**
    `AttackHypothesis` $\zeta \leftarrow$ GetSatisfyingAssignment(C)
    (`TestResult`, `EdgeResidues`) $\leftarrow$ EmbeddabilityTest($\hat{\eta}(N) \setminus \zeta$)
    if `TestResult` = `IsEmbeddable` **then**
      return ζ ▷ $\eta \preceq \zeta$
    else
      `NewC` $\leftarrow$ GenCounterexamples ($\hat{\eta}(N) \setminus \zeta$, `EdgeResidues`)
      $C \leftarrow C \cup$ `NewC` $\cup (\bigvee_{(i,j)\in\zeta} b_{ij} \vee \bigvee_{k\in\zeta} c_k)$ ▷ C on next iteration includes counterexamples and ζ
  return Failure

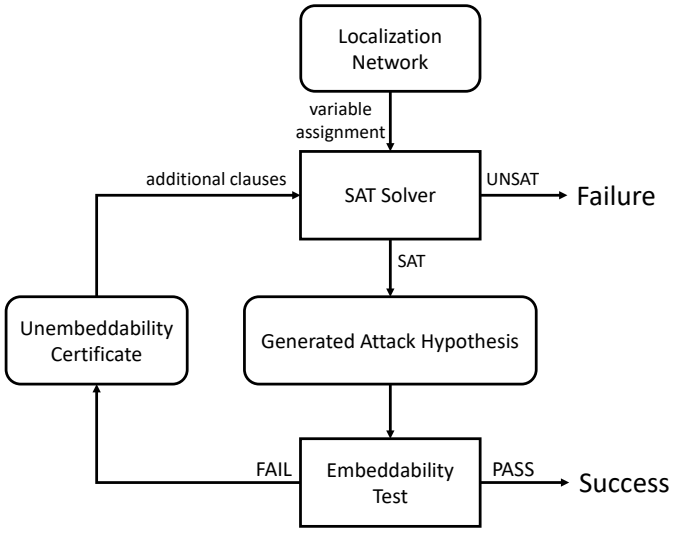

Figure 6.4: An illustration of the steps and overall flow of Algorithm 1

6.3 The Gordian Algorithm

GORDIAN is an algorithm designed to efficiently perform attack detection on finite localization networks, guaranteed to succeed on $(\overline{s}, \overline{t})$-AT localization networks. GORDIAN's design is inspired by Imhotep [86], Shoukry et al.'s lazy SMT solver for secure state estimation in the presence of attacks. Like Imhotep, GORDIAN identifies a combinatorial attack identification sub-problem that can be isolated from an otherwise convex optimization problem.

High Level Design

We present GORDIAN's high level process for solving the attack detection Problem 6.2.1 in Algorithm 1. GORDIAN's main steps are also graphically depicted in figure 6.4, and summarized here.

Given localization network N, GORDIAN first assigns a boolean variable to represent the corrupt/clean status for each of N's distance measurements and anchor positions. Next, GORDIAN assembles clauses made up of these variables to form a boolean satisfiability (SAT) problem,[3] in such a way that a satisfying assignment to the variables represents a plausible attack hypothesis. GORDIAN solves the SAT problem and tests the attack hypothesis ζ obtained as the SAT solution. This is accomplished by testing the embeddability of $N' =$

[3]Technically the constraints on line 4 of Algorithm 1 make this a pseudo-Boolean satisfiability problem, that is translatable into a boolean SAT problem.

$N \setminus \zeta$, a network with the hypothesized attacked edges removed. If N' is embeddable, by Lemma 6.2.4 all remaining attacks are trivial and ζ may be returned as the Problem 6.2.1 solution. If N' is not embeddable, GORDIAN attempts to find small $N'' \subseteq N'$ that are also not embeddable. Algorithm 2 attempts to find these N'' and uses them to construct new SAT counterexample clauses which direct future iterations of GORDIAN to test attack hypotheses containing edges and anchors from N''. By Theorem 6.7.1 in the Appendix, assuming a well-posed input, GORDIAN will always terminate in the "Success" zone of figure 6.4 with a Problem 6.2.1 solution.

We walk through an iteration of GORDIAN's main loop on the long-running example[4] in figure 6.3. GORDIAN first assigns boolean variables to the anchors a_1, a_2, a_3, a_4, a_5, a_6 and the edges $e_{(1,2)}$, $e_{(1,3)}$, ... $e_{(4,7)}$. In our convention, the literal x refers to the positive occurance of variable x and x' as its negation. The clauses, C are initialized with $e_{(1,2)} + e_{(1,3)} + \ldots\ e_{(4,7)} \leq \overline{s} = 1$ and $a_1 + a_2 + a_3 + a_4 + a_5 + a_6 \leq \overline{t} = 0$ to encode the assumption there are no more than $\overline{s}$ corrupted edges and no more than $\overline{t}$ corrupted anchors. The SAT solver immediately finds a satisfying assignment by setting every variable to false, signifying an empty attack hypothesis ζ. No edges or anchors are removed from $\hat{\eta}(N)$ on this iteration. An embeddability test is performed, and due to the inconsistency on edges connected to node 7, it fails. GORDIAN goes to the Trilateration Counterexamples (GENCOUNTEREXAMPLES) step starting with a randomly selected high residue edge, which happens to be (1,2). GORDIAN randomly selects edges (1,6) and (2,6) to complete the initial triangle. The induced sublocalization network for nodes 1, 2, and 6 is embeddable, so GORDIAN randomly selects node 7 to extend the counterexample. The induced sublocalization network for nodes 1, 2, 6, and 7 is still embeddable so GORDIAN tries again by randomly selecting node 3. Finally, the induced sublocalization network for nodes 1, 2, 3, 6 and 7 is *not* embeddable, so GORDIAN learns the clause $(a_1 \vee a_2 \vee a_3 \vee a_6 \vee e_{(1,2)} \vee e_{(1,3)} \vee e_{(1,6)} \vee e_{(1,7)} \vee e_{(2,3)} \vee e_{(2,6)} \vee e_{(2,7)} \vee e_{(3,6)} \vee e_{(3,7)})$, which contains every edge measurement and anchor position in the counterexample. On the next iteration of SAT solving, the new clause will lead to an attack hypothesis that sets at least one of those variables to true, shrinking the search space for the true attack.

Attack Hypothesis Generation

Assume we begin with a finite $(\overline{s}, \overline{t})$-AT localization network N that is corrupted by $\hat{\eta} \in \hat{H}(\overline{s}, \overline{t})$ (denoted $\hat{\eta}(N)$) as input. To model an attack hypothesis ζ on $\hat{\eta}(N)$, GORDIAN assigns pseudo-boolean indicator variables b_{ij} and c_k to the edges and anchors of its input. The attack hypothesis is a tuple of the edges and anchors for which the pseudo-boolean variables were set to 1. More formally $\zeta = (\{e_{ij} : b_{ij} = 1\}, \{a_k : c_k = 1\})$. This model allows attack hypotheses to be generated by a pseudo-boolean satisfiability (SAT) solver; initially only given the constraint there are fewer than $\overline{s}$ and $\overline{t}$ attacks, but over time accumulating counterexample clauses learned from EMBEDDABILITYTEST and GENCOUNTEREXAMPLES.

[4]The example in figure 6.3 is just 1-GGR and not technically a well-formed input for $\overline{s} = 1$. GORDIAN can still run on such a localization network, just without the guarantee of correctness from Theorem 6.2.6.

Embeddability Test

We will show in this section how to frame the embeddability (and localization) problem as a convex program, relying on a key result in the literature [90]: Uniquely Localizable (UL) localization networks with no attacks can always be localized in the plane using the (BY-SDP) technique [13] presented below. We take the contrapositive of this result to identify inconsistent localization networks.

Let the unknown $p(i)$ for $i \in S_x$ be decision variables and define $X = [p(1), p(2), ..., p(|S_x|)] \in \mathbb{R}^{2\times|S_x|}$ as the matrix of decision variables obtained by stacking the first and second coordinates of the $p(i)$. Also, let $\nu_i \in \{0,1\}^n$ be a unit column vector whose ith component is 1 and all other components 0, and $a_j \in \mathbb{R}^2$ be the position of anchor node j.

We define $g_{ij} = \begin{bmatrix}\nu_i - \nu_j & 0\end{bmatrix}^\top$ if both s_i and s_j are sensors, and $g_{ij} = \begin{bmatrix}\nu_i & -a_j\end{bmatrix}^\top$ if either of s_i and s_j is an anchor. Now, the squares of sensor-sensor distances and sensor-anchor distances can be uniformly represented as

$$\|p(i) - p(j)\|^2 = \left| g_{ij}^\top \begin{bmatrix}\mathbf{X}^\top\mathbf{X} & \mathbf{X}^\top \\ \mathbf{X} & \mathbf{I}_2\end{bmatrix} g_{ij} \right| \tag{6.3.1}$$

where $\mathbf{I}_2$ is a 2×2 identity matrix. With this representation for the aggregate $\|p(i) - p(j)\|^2$, we can set up an optimization problem of the form in (6.1.1). It is worthy to note that, the sensor set and the anchor set vary with the attack hypothesis. A hypothetically attacked anchor will be treated as a sensor with unknown location in the optimization problem.

$$\begin{aligned} \underset{\mathbf{X},\mathbf{Y}}{\text{minimize}} \quad & \sum_{(i,j)\in E} \underbrace{\frac{\left| g_{ij}^\top \begin{bmatrix}\mathbf{Y} & \mathbf{X}^\top \\ \mathbf{X} & \mathbf{I}_2\end{bmatrix} g_{ij} - W(e_{ij})^2 \right|}{W(e_{ij})^2}}_{residue_{(i,j)}} \\ \text{subject to} \quad & \mathbf{Y} = \mathbf{X}^\top\mathbf{X}. \end{aligned} \tag{6.3.2}$$

We observe that the objective in Problem 6.3.2 is a summation over the residues on each edge (i, j) which will attain a minimum of zero, i.e. all residues are zeros, if there exists an embedding. However, Problem 6.3.2 is not convex, because the constraint $\mathbf{Y} = \mathbf{X}^\top\mathbf{X}$ restricts the rank of $\mathbf{Y}$ to be 2.

Biswas and Ye's solution [13] to this dilemma is to lift the feasible set to a higher dimension by relaxing the constraint to $\mathbf{Y} \succeq \mathbf{X}^\top\mathbf{X}$ [12]. This yields the following SDP[5] solvable in polynomial time by interior point methods [14]. We refer to this relaxation as the BY-SDP.

$$\begin{aligned} \underset{\mathbf{X},\mathbf{Y}}{\text{minimize}} \quad & \sum_{(i,j)\in E_x} \frac{|g_{ij}^\top \mathbf{Z} g_{ij} - W(e_{ij})^2|}{W(e_{ij})^2} \\ \text{subject to} \quad & \mathbf{Z} \doteq \begin{bmatrix}\mathbf{Y} & \mathbf{X}^\top \\ \mathbf{X} & \mathbf{I}_2\end{bmatrix} \succeq 0. \end{aligned} \tag{6.3.3}$$

[5]The normalizing factor $W(e_{ij})^2$ in the denominator is not explicitly given by Biswas and Ye, but it is implicitly equivalent to the arbitrary multiplicative weights in [13].

Conceptually, this relaxation allows the solver to localize each sensor in a higher-dimension space $\mathbb{R}^{|S_x|}$ instead of in $\mathbb{R}^2$ [13]. If the residues are all zero and $\mathbf{Y}$ has rank 2, up to some numerical errors, we can certify that the localization network is embeddable in $\mathbb{R}^2$. If localization is desired, the component of $\mathbf{Z}$ corresponding to $\mathbf{X}$ can be read off as the projection of the high dimensional solution back down to the plane of the anchors.

Trilateration Counterexamples

Without an efficient way to reduce the search space of attack hypotheses, Algorithm 1 reduces to a brute force search over all $\binom{|E|}{|\bar{s}|} \cdot \binom{|S_a|}{|\bar{t}|}$ candidates for ζ. But if Algorithm 1 finds $\hat{\eta}(N) \setminus \zeta$ unembeddable, often only a small $\hat{\eta}(N') \subset \hat{\eta}(N) \setminus \zeta$ causes unembeddability. If Algorithm 2 can find an $\hat{\eta}(N')$, Algorithm 1 can reduce its search space dramatically by learning a counterexample clause constructed by taking the "or" of the boolean variables for nodes and edges in $\hat{\eta}(N')$. With the new clause, all attack hypotheses on future iterations must offer an explanation of why N' was unembeddable, focusing the search.

Our heuristic approach, presented in Algorithm 2, for finding small $\hat{\eta}(N')$ is motivated by the observation that high residue values from localization tend to occur in the *vicinity* of the attacks in the graph. As clean trilateration localization networks are GGR, Eren et al.'s iterative trilateration method [23] is the basis for Algorithm 2. Starting with three nodes, Algorithm 2 extends a candidate $\hat{\eta}(N')$ one node at a time until $\hat{\eta}(N')$ is determined to be unembeddable[6] or a maximum subgraph size (8 nodes in our implementation) is reached. We memorize calls to the EmbeddabilityTest procedure to save runtime.

ImportanceSamplingByResidue produces a starting point for a trilateration subgraph by weighting every edge in the graph by its residue, normalizing weights into a probability distribution summing to 1, and then sampling an edge from that distribution. ImportanceSamplingByConnectingResidues is a similar algorithm that samples nodes with at least three connections to the current trilateration subgraph. The weight assigned to a node is equal to the sum of edge residues on edges connecting it to the trilateration subgraph.

6.4 Noisy Gordian

Real world sensors are imperfect, introducing small errors into their measurements which shouldn't qualify as attacks. We now consider the Noisy Localization and Noisy Attack Detection problems where small errors are expected on uncorrupted values. With an extension of Gordian's embeddability test (section 6.3) to this noisy case, we show how Gordian can be used to identify violations of sensor noise assumptions.

[6] Non-zero BY-SDP residues are sufficient for showing unembeddability.

Algorithm 2 Trilateration Counterexamples

procedure GENCOUNTEREXAMPLES(localization network N, `EdgeResidues`)
  $C \leftarrow \emptyset$ ▷ C is the set of counterexamples
  for 1 : `NumberOfIterations` **do** ▷ `NumberOfIterations` is a parameter
    $(i, j) \leftarrow$ IMPORTANCESAMPLINGBYRESIDUE(`EdgeResidues`)
    `ThirdNodeCandidates` $\leftarrow \{k \ : \ k \in S \ \& \ (i,k), (j,k) \in E\}$
    $k \leftarrow$ IMPORTANCESAMPLINGBYCONNECTINGRESIDUES(`ThirdNodeCandidates`, `EdgeResidues`)
    $N' \leftarrow$ NEWSUBLOCALIZATIONNETWORKINDUCEDBY$_N(\{i, j, k\})$
    while $|S'| <$ `MaxSubgraphSize` **do** ▷ `MaxSubgraphSize` is a parameter
      if EMBEDDABILITYTEST$(N') =$ `NotEmbeddable` **then**
        $C \leftarrow C \cup (\bigvee_{(i,j)\in E'} b_{ij} \vee \bigvee_{k \in S'_a} c_k)$ ▷ At least one edge or anchor in N' is corrupted
        break
      `NextNodeCands` $\leftarrow \{n \ : \ n \in S, \exists \ i, j, k \in S'$ s.t. $(i, n), (j, n), (k, n) \in E\}$
      `n` $\leftarrow$ IMPORTANCESAMPLINGBYCONNECTINGRESIDUES(`NextNodeCands`, `EdgeResidues`)
      $N' \leftarrow$ NEWSUBLOCALIZATIONNETWORKINDUCEDBY$_N(S' \cup n)$
  return C

Noisy Localization Networks

To review our previous terminology, for localization network $N = (S, E, W, p_a)$, the Exact Localization Problem 6.1.2 presupposes the weights $W(e_{ij}) = d_{ij}$ and anchor positions $p_a = p^* \restriction_{S_a}$. An Attack Profile $\hat{\eta} = (b, c, m, a)$ substitutes $m_{ij} = d_{ij} * (1 + \delta_{ij})$ in place of weight $W(e_{ij})$ for $e_{ij} \in b$ and anchor positions $a_i = p^*(i) + \alpha_i$ for $p_a \in c$.

We now introduce Noise Profiles $\hat{\nu} = (b, c, m, a)$ as a specialized form of Attack Profile, where δ_{ij} and α_i are not arbitrarily determined by an adversary, but instead drawn from zero-mean probability distributions as $\delta_{ij} \sim \Theta_{ij}$ and $\alpha_i \sim \Phi_i$ with the symbol $\sim$ standing for the "is distributed according to" relation. A noise profile may be applied to a noiseless localization network N as $\hat{\nu}(N)$ and may be composed with an (adversarial) attack profile $\hat{\eta}$ to represent a both corrupted and noisy localization network as $\hat{\eta}(\hat{\nu}(N))$.

Since our noise assumptions give anchor positions the possibility of error (like in GNSS sensors), we give the "softened" anchor constraints version of Equation 6.1.1. The new constant λ weights the relative significance of distance measurements against anchor positions in the objective. In this chapter we, somewhat arbitrarily, use $\lambda = 5$.

$$\underset{p(i), i \in S}{\text{minimize}} \sum_{(i,j)\in E} \frac{|\|(p(i) - p(j)\|_2^2 - W(e_{ij})^2|}{W(e_{ij})^2} + \lambda \sum_{k \in S_a} \|(p(k) - p_a(k)\|_2^2. \tag{6.4.1}$$

Observe the change from S_x and E_x in Equation 6.1.1 to S and E in the first summation of Equation 6.4.1. S changes because anchor positions are now decision variables. E changes

because inter-anchor measurements are useful when anchors can "float". When anchor positions were hard constraints, the inter-anchor distance measurements taken by sensors were irrelevant and hence excluded via E_x.

Approximate Embeddings

We cannot simply extend Problem 6.1.2 to the noisy case by treating $\hat{\nu}(N)$ as input in place of N, because even attack-free noisy localization networks are almost never consistent. As Anderson et al. observe, solving Problem 6.1.2 for a noisy GGR localization network is equivalent to finding the solution to an over-determined system of polynomial equations (the distance constraints), and such solutions do not in general exist [4]. Therefore we define noisy localization as the solution to the optimization problem in Equation 6.1.1.

Problem 6.4.1 (Noisy Localization)**.** *Given $\hat{\eta}(N)$ for localization network N and noise profile $\hat{\eta}$, find the solution to* (6.1.1)*.*

Framing noisy localization directly as an optimization problem is common in the literature (e.g. [12, 82, 96]). In this chapter as we consider anchor noise, we instead use Equation 6.4.1 and solve the similar Problem 6.4.2.

Problem 6.4.2 (Noisy Localization with Soft Anchors)**.** *Given $\hat{\eta}(N)$ for localization network N and noise profile $\hat{\eta}$, find the solution to* (6.4.1)*.*

Problems 6.4.2 and 6.4.1 are not given in terms of finding p^*; however Anderson et al. show (a statement similar to) Problem 6.4.1 has useful properties such as a unique minimum for a uniquely localizable N when noise is sufficiently small [4]. Therefore a solution to Problems 6.4.2 and 6.4.1 can be thought of as an *approximate* solution to Problem 6.1.2.

We seek to generalize the notion of a noiseless embedding to an *approximate* embedding in an analogous way by examining residues. As a noiseless embedding has zero residues, an approximate embedding should have low residues. But how low?

Consider $N = (S, E, W, p_a)$ and a ground truth placement p^* s.t. $W(e_{ij}) = d_{ij}$ and $p_a = p^* \restriction_{S_a}$. In the remainder of this section we explain how to set residue thresholds for approximate embeddability in such a way that p^* is an approximate embedding of $\hat{\nu}(N)$ at high confidence in the absence of an attack. If some N' were not approximately embeddable in this way, with high confidence $N' \neq \hat{\nu}(N)$, implying N' contains an attack.

We have already defined an edge $residue_{(i,j)}$ from Equation 6.3.2. Now with noise on anchors we may also define an anchor $residue_i$ as $\|(p(k) - p_a(k)\|_2^2$. Note for both edges and anchors, residues are defined with respect to a particular placement p.

Definition 6.4.3 (γ-β-Embedding)**.** *A placement p is a γ-β-Embedding with respect to localization network N if*

1. $\sum_{(i,j)\in E} residue_{(i,j)} \leq \gamma_e$.

2. $\sum_{k \in S_a} residue_k \leq \gamma_a$.

3. $\forall (i,j) \in E\ residue_{(i,j)} \leq \beta_{ij}$.

4. $\forall k \in S_a\ residue_k \leq \beta_k$.

Conditions (1) and (2) appear in the the objective function for noiseless embeddability testing with soft anchor constraints in Equation 6.4.1. The β parameter used for conditions (3) and (4) captures the idea that bounded noise distributions Θ and Φ should yield bounded residues. The choice of particular conditions for Definition 6.4.3 was motivated by the following observations about the residues at p^*:

Since $d_{ij}^2 = \|(p^*(i) - p^*(j)\|_2^2$, and d_{ij}^2 appears in every term of noisy $W(e_{ij})$, it can be factored out, which yields

$$\begin{aligned} residue_{ij} = \frac{\left|d_{ij}^2 - m_{ij}^2\right|}{m_{ij}^2} &= \frac{\left|d_{ij}^2 - (d_{ij}^2(1+\delta_{ij})^2)\right|}{d_{ij}^2(1+\delta_{ij})^2} \\ &= \frac{|1 - (1+\delta_{ij})^2)|}{(1+\delta_{ij})^2} \end{aligned} \tag{6.4.2}$$

The same is true for $p^*(i)$ in

$$residue_k = \|p^*(k) - (p^*(k) + \alpha_k)\|_2^2 = \|\alpha_k\|_2^2. \tag{6.4.3}$$

These factored residue expressions contain no instances of d_{ij} nor p^*, yet represent the value of residues when $p = p^*$. Therefore, *it is not necessary to know p^* to calculate the residues at p^*.* Noisy p^* residues may be determined directly from Θ and Φ.

These observations enable the selection of γ and β values G and B such that p^* is G-B-embeddable at high confidence. Monte Carlo simulation is a simple and effective method for settling on G and B values appropriate for Θ and Φ. The algorithm is straightforward: for a sufficiently large number of trials, sample δ and α values and compute $\sum_{(i,j) \in E} residue_{(i,j)}$ and $\sum_{k \in S_a} residue_k$. Looking at the histogram of residue sums across the trials, pick a high percentile (e.g. the 99.9th percentile) and use the corresponding values for G_e and G_a. The same algorithm works for B_e and B_a, and as we show in Section 6.5 is applicable to arbitrary Θ and Φ such as data sets from real-world sensors.

Approximate Embeddability Test

We next show how an SDP formulation for noisy localization may be adapted to G-B-embeddability testing via a constraint satisfaction problem (CSP) with a semidefinite relaxation.

The BY-SDP procedure (6.3.3) was originally intended as a solution to Problem 6.4.1 for noisy localization networks and is empirically effective with noisy measurements [12]. We give the soft-anchor variant of the method for Problem 6.4.2 in Equation 6.4.4. This

alternative formulation can be intuitively understood as treating anchor nodes as unlocalized nodes with an anchor residue term in the objective penalizing the anchor's distance from its nominal position. Formally, a modification must be made to matrix $\mathbf{X}$ by including anchor nodes as decision variables. Here $\mathbf{X} = [p(1), p(2), ..., p(|S|)] \in \mathbb{R}^{2\times|S|}$. Let $g_{ij} = \begin{bmatrix}\nu_i - \nu_j & 0\end{bmatrix}^\top$ *for all* s_i and s_j. Separate the other case of g from before to $f_i = \begin{bmatrix}\nu_i & -a_j\end{bmatrix}^\top$ for $s_k \in S_a$. The definitions of $\mathbf{Y}$ and $\mathbf{Z}$ are as before, but with respect to the new $\mathbf{X}$.

$$\underset{\mathbf{Z}\succeq 0}{\text{minimize}} \sum_{(i,j)\in E} \frac{|g_{ij}^\top \mathbf{Z} g_{ij} - W(e_{ij})^2|}{W(e_{ij})^2} + \lambda \sum_{k\in S_a} f_k^\top \mathbf{Z} f_k \tag{6.4.4}$$

The first summation expresses inter-node residues changed to sum over E in place of E_x because inter-anchor measurements provide information when anchor positions are not hard constraints. The second summation expresses anchor residues ($\|(p(i) - p_a(i)\|_2^2$). As before, the semidefinite relaxation of Z changes this from an nonconvex rank-constrained problem to an SDP problem. To obtain the location estimate, read $\mathbf{X}^*$ off the minimizer $\mathbf{Z}^*$.

The important thing about Equation 6.4.4 from an approximate embeddability testing perspective is that it contains expressions for each edge and anchor residue. This is everything we need to test the G-B-embeddability of network N:

$$\sum_{(i,j)\in E} \frac{\left|g_{ij}^\top \mathbf{Z} g_{ij} - W(e_{ij})^2\right|}{W(e_{ij})^2} \leq G_e \tag{6.4.5a}$$

$$\sum_{k\in S_a} f_k^\top \mathbf{Z} f_k \leq G_a \tag{6.4.5b}$$

$$\frac{\left|g_{ij}^\top \mathbf{Z} g_{ij} - W(e_{ij})^2\right|}{W(e_{ij})^2} \leq B_{ij}, \forall (i,j) \in E \tag{6.4.5c}$$

$$f_k^\top \mathbf{Z} f_k \leq B_k, \forall k \in S_a, \tag{6.4.5d}$$

If the above constraints are not feasible with respect to any $\mathbf{Z} \succeq 0$, we can then conclude that the given N is not G-B-embeddable; however, if the constraints are feasible, we cannot be sure whether N is truly G-B-embeddable in 2 dimensions or simply appears as such due to the relaxation of the rank constraint on Z.

Noisy Attack Detection

Noisy attack detection generalizes from the noiseless case.

Problem 6.4.4 (Noisy Attack Detection)**.** *For a localization network N, noise profile $\hat{\nu}$, and attack profile $\hat{\eta}$, given only $\hat{\eta}(\hat{\nu}(N))$, find a hypothesis $\zeta \in H(\overline{s}, \overline{t})$ such that $\hat{\eta}(\hat{\nu}(N)) \setminus \zeta$ is G-B-embeddable.*

We implemented Noisy Gordian by replacing the embeddability test in section 6.3 with Monte Carlo calculation of G and B and the CSP[7] in Equation 6.4.5. Noisy Gordian solves Problem 6.4.4 up to the SDP relaxation of its embeddability test. By the selection of parameters G and B, such a hypothesis exists (when $\zeta = \hat{\eta}$) with high confidence for $\hat{\eta}(\hat{\nu}(N)) \setminus \zeta$, and lines 7 and 8 of Algorithm 1 ensure Noisy Gordian will not terminate until it has found one. We classify non-trivial attacks (larger in magnitude than plausible noise from $\hat{\nu}$) into two categories: *G-B-effective* and *G-B-compatible*.

- **G-B-Effective Attacks**: $\hat{\eta}$, yielding G-B-unembeddable $\hat{\eta}(\hat{\nu}(N))$.
- **G-B-Compatible Attacks**: non-trivial $\hat{\eta}$, yielding G-B-embeddable $\hat{\eta}(\hat{\nu}(N))$.

When Noisy Gordian solves Problem 6.4.4 it identifies G-B-effective attacks. Any unidentified G-B-compatible attacks are limited in size by the uncorrupted part of the localization network: If p is a G-B-embedding of $\hat{\eta}(\hat{\nu}(N))$, p must also be a G-B-embedding of $\hat{\eta}(\hat{\nu}(N)) \setminus \hat{\eta}$. This is obvious with respect to definition 6.4.3 where all conditions true for the residues of $\hat{\eta}(\hat{\nu}(N))$ must also hold for the subset of those residues in $\hat{\eta}(\hat{\nu}(N)) \setminus \hat{\eta}$. Therefore "$G$-$B$-compatible" attacks can only be as bad as the worst G-B-embeddable p of $\hat{\eta}(\hat{\nu}(N)) \setminus \hat{\eta}$, which the adversary does not influence.

Put simply, for geometric configurations or noise conditions where you wouldn't expect a "good" noisy localization result, you shouldn't expect a "good" attack detection result (i.e with limited G-B compatible attacks). In the world of navigation systems, Geometric Dilution of Precision (GDP) [48] describes how the geometry of noisy range measurements can affect localization precision for a single sensor. Even for GGR localization networks, poor geometry on measurements may result in poor localization precision and poor attack detection. The theoretical aspects of this subject are worthy of further research, but for now we estimate the role of geometry and network topology empirically in our experimental evaluation by searching for maximum size G-B-compatible attacks on our benchmarks.

6.5 Results

As the correctness of Gordian and related algorithms are proved in the previous sections, our empirical evaluation is focused on evaluating performance. In this section we evaluate the runtime of noiseless and noisy Gordian.

Experimental Setup

We implemented Gordian in Matlab 2016b, using the Yalmip toolbox [63] to model the Semidefinite Programming (SDP)s. Our implementation uses MOSEK [70] and SAT4J's

[7](6.4.5) is a constraint satisfaction problem, and does not produce meaningful residues as expected by Line 7 of Algorithm 1 in equation 6.4.5. As a workaround we generate residues by first solving a noisy localization problem (6.4.4) whenever residues are required.

BM	#Nodes	#Edges	$\overline{s}$	$\overline{t}$	BY-SDP (s)	ESDP (s)	$+\delta'_{ij}$	$-\delta'_{ij}$	$\|\alpha_i\|$	Noisy (s)	Correct	Nearly Correct
1	20	176	1	2	65.87	81.67	2.1	2.8	1.6	142.40	100%	0%
2	20	170	2	2	73.54	89.65	2.2	1.9	1.4	199.23	100%	0%
3	20	170	2	3	66.22	64.74	1.9	2.0	1.5	197.77	100%	0%
4	20	160	1	3	61.01	50.87	4.5	*	1.5	152.96	80%	20%
5	20	171	4	2	74.96	91.53	2.8	1.7	1.7	354.38	60%	0%
6	20	167	4	3	74.06	93.39	1.9	2.9	1.7	*	0%	40%
7	30	369	4	3	163.39	208.32	2.4	2.3	1.6	480.68	100%	0%
8	30	397	7	0	2140.79	211.23	2.1	1.8	*	218.74	80%	0%
9	30	411	6	0	104.62	174.55	2.8	2.2	*	446.83	80%	0%
10	30	381	6	2	190.13	259.04	1.9	2.1	1.8	811.40	80%	0%
11	30	368	0	5	46.00	76.81	*	*	1.6	142.25	100%	0%
12	30	372	2	7	160.67	201.96	2.3	2.0	1.4	645.57	80%	0%
13	40	716	5	7	1186.59	454.86	2.2	2.0	1.5	2671.00	100%	0%
14	40	723	5	3	267.82	382.80	1.9	2.0	1.6	1786.09	100%	0%
15	40	501	2	2	145.65	142.42	2.0	1.9	1.6	422.64	100%	0%
16	60	908	2	2	404.21	303.91	2.0	3.6	1.8	1398.83	60%	40%
17	70	1070	2	2	722.18	754.97	2.0	1.3	1.7	1383.09	80%	0%
18	80	1258	2	2	366.71	319.39	2.5	*	1.6	969.74	80%	0%

Table 6.1: GORDIAN runtime on BenchMarks (BM) using either BY-SDP or ESDP embeddability tests in the noiseless case, maximum undetectable attacks in the noisy case, and noisy runtime with accuracy. Noiseless runtimes are averages over 6 trials. For comparison, a brute-force (no trilateration counterexamples) BY-SDP trial took 3193.20 seconds on benchmark 1 and over 72 hours on benchmark 2 without terminating. δ' and α values represent maximum undetectable attacks. A * in the table is a placeholder for an impossible to collect value, either because there were no attacked edges/anchors in the benchmark or all attacked edges in the benchmark were too short to determine an effective negative attack. Noisy runtimes are averages over the fully correct runtimes found over 5 trials. Nearly correct trials had at most 1 misidentified attack. Benchmark 6 had no fully correct (all attacks identified) runtimes and hence no data point. Our explanation for this is below.

pseudoboolean solver [52] as the underlying SDP and Boolean Satisfiability Problem (SAT) solvers. Our testing platform is a Windows machine with a quad-core Intel Core i7-4700MQ CPU and 16GB memory.

We generated benchmark graphs by randomly dropping nodes onto a 15 by 15 grid, and connecting those points with inter-node distances within a fixed connection radius[8]. Attacks consist of $\overline{s}$ randomly selected edges and $\overline{t}$ randomly selected anchors. In each benchmark, $2\overline{t}+3$ nodes are randomly determined to be anchors.

We model Θ with data obtained from Lab 11, the authors of the SurePoint Localization System [44]. SurePoint uses DecaWave DW1000 ultra-wideband transceivers to achieve highly accurate ranging data in a real world enviornment. SurePoint achieves a median *additive* error of 0.08m with -0.59m and 0.31m as 95th percentile errors after proccessing.

A noise profile $\hat{\nu}$ was determined in the following way. Anchors were moved in a uniformly

[8]Due to scaling by $W(e_{ij})$ in the denominator of edge residues, a short edge with large noise can overpower the other residues. To prevent this problem in our evaluation we enforce a minimum measurement length of 1m before noise and 0.3m after noise.

random direction a distance sampled from a truncated normal distribution clipped at 1.0 with mean 0 and standard deviation 0.2. Bounded additive noise was sampled uniformly at random from the SurePoint data set after eliminating the 405 out of 73414 outlier[9] values of noise with absolute error value greater than 0.7. We used these thresholds (1.0 and 0.7) to calculate B_{ij} and B_i values and Monte Carlo simulation to determine the 99.9th percentile of residue sums to set G_e and G_a values for G-B-Embeddability and Noisy Gordian experiments.

Noiseless Gordian

Our goal in experimental evaluation of the Noiseless GORDIAN algorithm is to evaluate the efficiency of GORDIAN with noiseless embeddability tests on localization networks with random attacks. The result for each benchmark is listed in Table 6.1.

In addition to the BY-SDP noiseless embeddabilty test, we also evaluate a (potentially faster) noiseless embeddability test built from Wang et al.'s alternative *edge*-centric (ESDP) relaxation of equation 6.3.2. Like the BY-SDP's rank condition for embeddability, Theorem 4.2 from [96] asserts that in the absence of attack, when the diagonal elements of $\mathbf{Y} - \mathbf{X}^\top\mathbf{X}$ are zero, all nodes have been localized to their true locations. Since an unembeddable localization network cannot simultaneously localize nodes to their true locations and achieve zero residues, the combination of zero residues and zero $trace(\mathbf{Y} - \mathbf{X}^\top\mathbf{X})$ certifies an embeddable ESDP solution.

In our trials of GORDIAN, BY-SDP vs ESDP embeddability tests usually result in roughly comparable runtimes, in line with the expectation of the SDP trending faster on dense localization networks with fewer nodes and the ESDP trending faster on sparse localization networks with more nodes [96]. Benchmarks 8 and 13 show instances where the ESDP achieved a much faster runtime than the BY-SDP. This may suggest in these cases a difference in the utility of ESDP residues and BY-SDP residues for trilateration counterexample generation.

Numerical errors and solver inaccuracies were significant challenges for implementation of embeddability tests in our noiseless experiments where picking thresholds for "nearly zero" and "nearly rank two" required fine-tuning. However these concerns are dwarfed by measurement error in the noisy case, which we consider for the remainder of our evaluation.

G-B-Embeddability

We identified two options for relating additive SurePoint noise ($m_{ij} = d_{ij} + \delta'_{ij}$) to the multiplicative noise model ($m_{ij} = d_{ij}(1 + \delta_{ij})$) assumed for Equation 6.4.2: obtain an equivalent $\delta_{ij} = \delta'_{ij}/d_{ij}$ or directly estimate d_{ij} as $d_{ij} = m_{ij} - \delta'_{ij}$ to calculate residue estimates from $residue_{ij} = \left|d_{ij}^2 - m_{ij}^2\right|/m_{ij}^2$ and the m_{ij} values on hand. We use this last approach in our evaluation to determine G values from Monte Carlo simulation.

[9] We eliminate outliers from the noise distribution to ensure that the only outliers in our experiments are the attacks we deliberately apply to our benchmarks.

The forth section of Table 6.1 addresses the performance of the G-B embeddability test[10] from equation 6.4.5. As discussed in Section 6.4, noise levels and the particular geometry of a benchmark determine whether an attack is G-B compatible (not detectable) or G-B effective (detectable). As we can only evaluate the runtime of GORDIAN when it has effective attacks to detect, we first set out to estimate the minimum size of an effective attack on our benchmarks.

This was accomplished by starting with the attack free and noise free underlying N of a benchmark, applying SurePoint and anchor noise $\hat{\nu}(N)$, arbitrarily selecting a single attacked edge/attacked anchor and corrupting it with a positive edge attack ($m_{ij} = d_{ij} + \delta'_{ij}$), a negative edge attack ($m_{ij} = d_{ij} - \delta'_{ij}$), or an anchor attack ($a_i = p_a(i) + \alpha_i$) to produce $\hat{\eta}(\hat{\nu}(N))$. Starting at the noise bound we incrementally increased the magnitude of the attack ($|\delta'_{ij}|$ or $\|\alpha_i\|$) until the resulting $\hat{\eta}(\hat{\nu}(N))$ became not G-B-Embeddable. The results in the 4th section of table 6.1 are averages across 5 trials of this process. This methodology produced useful estimates for the minimum size of a G-B-effective attack in these benchmarks given our noise conditions.

Noisy Gordian

As discussed, noisy GORDIAN is only able to detect G-B-effective attacks. Therefore for each edge attack we set $\delta'_{ij} \in [5, 6]$ and for each anchor attack $\|\alpha_i\|_2 \in [5, 6]$ so all attacks would likely be over the threshold of G-B-effectiveness. Edge attacks were given a 50% chance of being positive or negative[11]. In this way $\hat{\eta}$ was fixed for each benchmark. We generated 5 noise profiles $\hat{\nu}$ and for each profile evaluated a trial of Noisy GORDIAN on each $\hat{\eta}(\hat{\nu}(N))$ of our benchmarks.

The final section of table 6.1 contains the results of this experiment. The average runtimes across trials yielding fully correct attack detection results are presented in the first column. This allows a nearly apples-to-apples comparison against runtimes in the noiseless case where correct attack detection results are always produced. Overall, Noisy GORDIAN is significantly slower than corresponding noiseless problems, but is still much faster than brute force.

In the majority of trials, Noisy GORDIAN successfully identified the full attack. When Noisy GORDIAN failed in our experiments, either it confused *no more than one* uncorrupted edge or anchor with a corrupted edge or anchor, or the G-B-embeddability test misclassified an embeddable sublocalization network as unembeddable. In the latter case, Noisy GORDIAN learns a counterexample clause that makes the SAT problem on line 5 of algorithm 1 unsatisfiable (which is always a risk considering the 99.9% confidence intervals on G). This was what happened with benchmark 6. By repeating the tests for benchmark 6 with a bigger (99.99%) confidence interval, the results improved to 0% correct and 80% nearly correct. In

[10] We implemented and evaluated an ESDP-style approximate embeddability test too, but determined the ESDP's weaker relaxation has too many false negatives, i.e. localization networks which only appear embeddable under the ESDP relaxation, to justify its use.

[11] If a negative attack is impossible because d_{ij} is small and edges cannot have negative length, it is changed to a positive attack

summary, *over all experiments, Noisy* GORDIAN *either produced a correct result, a nearly correct result with a single error, or failed while indicating no result could be found at all.*

6.6 Conclusion

We have presented GORDIAN, an attack detection algorithm at the distance-graph abstraction level. In the noiseless case, we proved GORDIAN sound and complete for $(\overline{s}, \overline{t})$-AT input. By generalizing localization network embeddability testing to approximate embeddability testing in the noisy case, we likewise extend GORDIAN to Noisy GORDIAN. We evaluate our algorithms on noisy attack detection problems constructed from the error distribution of a real-world localization system, demonstrating the practical utility of our formal reasoning based outlier detection scheme.

GORDIAN and Noisy GORDIAN consistently finished combinatorially difficult attack detection benchmarks with a high success rate on the order of minutes instead of days. These algorithms are appropriate as a periodic check to identify bad actors in a long-running sensor network or general network of cyber-physical systems. Once adversarial data are removed, localization can be made more trustworthy and more accurate. Dynamic IoT services may then have higher confidence in spatial models.

6.7 Appendix

Theorem 6.7.1. *Given a finite $(\overline{s}, \overline{t})$-AT localization network $N = (S, E, W, p_a)$ and a ground truth placement p^* s.t. $W(e_{ij}) = d_{ij}$ and $p_a = p^* \restriction_{S_a}$ and $\hat{\eta} \in \hat{H}(\overline{s}, \overline{t})$, Algorithm 1 is a sound and complete procedure for solving Problem 6.2 when given input $\hat{\eta}(N)$.*

Proof. (Soundness) The algorithm returns with an attack hypothesis $\eta_{N'}$ only if $\hat{\eta}(N')$ is consistent. But since N was $(\overline{s}, \overline{t})$-AT, that means by definition that $\hat{\eta}$ is trivial with respect to N', and so $\hat{\eta} \subseteq \eta_{N'}$.

(Completeness) Let η be the attack hypothesis induced by $\hat{\eta}$. If on line 4 we choose an $(\overline{s}, \overline{t})$-attack hypothesis $\zeta \succeq \eta$, $\hat{\eta}(N) \setminus \zeta$ is consistent and the algorithm ends. Denote by C_i the set of clauses at iteration i and let $SAT(C_i)$ be the set of satisfiable assignments to C on iteration i. Since C_0 simply bounds the number of attacks, clearly $\eta \in SAT(C)$. C_{i+1} is constructed $C_{i+1} \supseteq C_i$ (line 12) to forbid a reassignment on future iterations matching ζ (line 12 last term). Specifically $\zeta \in SAT(C_i)$, but $\zeta \notin SAT(C_{i+1})$, and $SAT(C_{i+1}) \subset SAT(C_i)$. $SAT(C_0)$ is finite, and so we just have to show that if $\eta \in SAT(C_i)$ and the algorithm doesn't end, $\eta \in SAT(C_{i+1})$.

On every iteration where the algorithm doesn't end we add clauses to C of the form $(\bigvee_{(i,j)\in \mathtt{E}} b_{ij} \vee \bigvee_{k\in \mathtt{Sa}} c_k)$ for some inconsistent sub-localization network. At least one of the b_{ij}, c_k must be in the true attack hypothesis η, or the sub-localization network would be consistent (the unattacked localization network was consistent, and removing edges or anchors can't make it less consistent). And so η satisfies the new added clause. □

Chapter 7

Conclusion

Truly successful technologies fade into the background. For instance the door is an amazing architectural innovation which saves people from punching a hole in their wall every time they want to step outside. But in the 21st century, very few people besides architects and locksmiths spend a lot of time thinking about doors. To the majority of homeowners and office dwellers, a door is only notable if there is something wrong with it. I believe the most positive future outcome for the IoT is for it to become as boring as a door.

Unfortunately, today's IoT is interesting in its failures. Security vulnerabilities, privacy violations, pointless products, and plain buggy technology frequently leave smart things worse than traditional dumb things. An example of how cloud-centric static IoT services can go wrong occurred in March 2019 when owners of the Jibo social robot lost access to most interactive features after its company went out of business.[1] The robot was designed to work exclusively with Jibo's servers which meant there was no alternative way for its owners to replace the lost features. Imagine if other industries worked this way; if for instance your furniture collapsed, or your food spoiled depending on the economic fortune of a company half-way around the world. Yet somehow the Jibo version of "ownership" is tolerated in the consumer IoT. If the Jibo robot had been designed as a dynamic IoT service, another cloud backend for its interactive features could have been substituted in for Jibo's dying servers.

I believe model based design of dynamic IoT services is an essential step toward eliminating these flaws. I also believe an edge-centric design which puts control of personal data and hardware at the hands of the user is the right approach for empowering consumers. This was the focus of chapter 3, which addressed the topic of IoT service composition. We presented the accessor project as a programming platform for coordinating the interaction of IoT systems in the fog instead of the cloud. Swarmlets with deterministic temporal semantics for service composition lead to more reliable and better testable implementations for both consumer and industrial applications. Downloadable components enable dynamic IoT services such as our augmented reality demonstration.

I suspect a main reason why IoT devices are usually designed as standalone stovepipe

[1] techcrunch.com/2019/03/04/the-nely-death-of-jibo-the-social-robot

systems is due to a weakness in spatial/contextual modeling. Why bother deploying a sophisticated indoor localization system in an environment if other systems can't interface with its spatial representation? We addressed this problem in chapter 2 where we surveyed existing models of space and observed that different representations are appropriate for different kinds of systems. Using concepts from model theory we proposed semantic localization as an interface between location representations and location programming.

When both contextual models of the world (semantic localization) and a platform for IoT composition (accessors) are available, we have the tools to build dynamic IoT services which use contextual information to find useful systems. We gave an architecture for designing such dynamic services in chapter 4 with a focus on the connected car. Our semantic accessors integrate the accessor project with the semantic web of things. Accessors both enhance the accessibility of semantic repositories and encapsulate the networking complexity of connecting to discovered services. As a prototype for a dynamic IoT service, we presented a dynamic dashboard user interface which automatically renders web component interfaces from discovered accessors.

Yet a dynamic user interface is not enough for addressing semantic interpretation for dynamic IoT services without a human in the loop. In chapter 5 we introduced special accessors called "adapters" to perform ontology matching across the spatial/contextual models in which sensors and systems interpret their data. We proved the theoretical validity of cross ontology inference under certain model-theoretic assumptions and showed how the semantic accessor architecture may be extended to discover contextually relevant adapters. We presented a demonstration of adapters via a robot following assistant which makes opportunistic use of a location sensor that was not originally engineered as part of the robot.

Dynamic IoT services have enormous potential due to their reliance upon spatial/contextual models of the world. But dangerous failure modes for services are possible if their models are incorrect or contain malicious data. We presented an algorithm for verifying and eliminating inconsistencies from spatial ontologies with sensor network distance measurements in chapter 6. Our algorithm, Gordian, uses the same techniques used to implement SMT solvers to rapidly identify inconsistent measurements. We determined theoretical conditions, based on rigidity theory, for guaranteed attack/outlier detection in the noiseless case. We extended Gordian to the noisy case by introducing and solving a noisy version of the graph-embeddability problem. Compared to a naive implementation, our algorithm can solve attack detection problems on the order of minutes instead of days.

Today's IoT is a paradox of extremes. The hype is enormous, the benefits may be huge, and the potential for critical security and privacy failings is intolerable. The research presented in this thesis is intended to help direct IoT design so the good may be attained and the bad may be avoided. Dynamic IoT services have the potential to leverage emergent properties of composition and contextual awareness to do incredible things. The IoT's future is coming, and with luck it will be as quietly excellent as a door.

Bibliography

[1] Joshua Adkins et al. "Demo: Michigan's IoT Toolkit". en. In: *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems - SenSys '15*. Seoul, South Korea: ACM Press, 2015, pp. 485–486. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2817866. URL: http://dl.acm.org/citation.cfm?doid=2809695.2817866 (visited on 04/26/2019).

[2] Gul Agha. *Actors: A Model of Concurrent Computation in Distributed Systems*. Cambridge, MA, USA: MIT Press, 1986. ISBN: 0-262-01092-5.

[3] Ilge Akkaya. "Data-Driven Cyber-Physical Systems via Real-Time Stream Analytics and Machine Learning". PhD thesis. EECS Department, University of California, Berkeley, Oct. 2016. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2016/EECS-2016-159.html.

[4] Brian D. O. Anderson et al. "Formal Theory of Noisy Sensor Network Localization". en. In: *SIAM Journal on Discrete Mathematics* 24.2 (Jan. 2010), pp. 684–698. ISSN: 0895-4801, 1095-7146. DOI: 10.1137/100792366. (Visited on 05/31/2016).

[5] Alexandre Armand, David Filliat, and Javier Ibañez-Guzman. "Ontology-Based Context Awareness for Driving Assistance Systems". In: *IEEE Intelligent Vehicles Symposium (IV)*. Dearborn, United States, June 2014, pp. 1–6. URL: https://hal.archives-ouvertes.fr/hal-01012078.

[6] Nikolay Atanasov et al. "Semantic localization via the matrix permanent". In: *Robotics: Science and Systems*. 2014. URL: http://roboticsproceedings.org/rss10/p43.pdf (visited on 04/21/2016).

[7] Prateek Bansal and Kara M. Kockelman. "Forecasting Americans long-term adoption of connected and autonomous vehicle technologies". In: *Transportation Research Part A: Policy and Practice* 95 (2017), pp. 49–63. ISSN: 0965-8564. DOI: https://doi.org/10.1016/j.tra.2016.10.013. URL: http://www.sciencedirect.com/science/article/pii/S0965856415300628.

[8] Payam M. Barnaghi et al. "Semantics for the Internet of Things: Early Progress and Back to the Future". In: *Int. J. Semantic Web Inf. Syst.* 8 (2012), pp. 1–21.

[9] Robert Battle and Dave Kolas. "Enabling the Geospatial Semantic Web with Parliament and GeoSPARQL". In: *Semant. web* 3.4 (Oct. 2012), pp. 355–370. ISSN: 1570-0844. URL: http://dl.acm.org/citation.cfm?id=2590208.2590211.

[10] Mark de Berg et al. *Computational Geometry: Algorithms and Applications*. 3rd ed. Santa Clara, CA, USA: Springer-Verlag TELOS, 2008. ISBN: 3540779736, 9783540779735.

[11] Tim Berners-Lee. *Linked Data - Design Issues*. July 2006. URL: https://www.w3.org/DesignIssues/LinkedData.html (visited on 12/08/2018).

[12] Pratik Biswas and Yinyu Ye. "Semidefinite programming for ad hoc wireless sensor network localization". In: *Proceedings of the 3rd international symposium on Information processing in sensor networks*. ACM, 2004, pp. 46–54. (Visited on 03/28/2016).

[13] P. Biswas et al. "Semidefinite Programming Approaches for Sensor Network Localization With Noisy Distance Measurements". In: *IEEE Transactions on Automation Science and Engineering* 3.4 (Oct. 2006), pp. 360–371. ISSN: 1545-5955. DOI: 10.1109/TASE.2006.877401.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[15] Christopher Brooks et al. "A Component Architecture for the Internet of Things". en. In: *Proceedings of the IEEE* 106.9 (Sept. 2018), pp. 1527–1542. ISSN: 0018-9219, 1558-2256. DOI: 10.1109/JPROC.2018.2812598. URL: https://ieeexplore.ieee.org/document/8343871/ (visited on 11/27/2018).

[16] H Butler et al. *The GeoJSON Format*. Tech. rep. July 2015. URL: https://www.ietf.org/archive/id/draft-butler-geojson-06.txt (visited on 01/03/2016).

[17] Gilberto Câmara, Frederico Fonseca, and Antonio Miguel Monteiro. "Algebraic structures for spatial ontologies". In: *II International Conference on Geographical Information Science (GIScience 2002), Boulder, CO, AAG*. 2002. URL: http://www.dpi.inpe.br/gilberto/papers/ontologies_giscience2002.pdf (visited on 01/03/2016).

[18] Anthony G. Cohn, Jochen Renz, et al. "Qualitative spatial representation and reasoning". In: *Handbook of knowledge representation* 3 (2008), pp. 551–596. URL: http://www.bibotu.com/books/2013/History%20and%20Philosophy%20of%20Science/Harmelen%20-%20Handbook%20of%20Knowledge%20Representation%20(Elsevier,%202008).pdf#page=577 (visited on 02/05/2016).

[19] Michael Compton et al. "The SSN ontology of the W3C semantic sensor network incubator group". In: *Web Semantics: Science, Services and Agents on the World Wide Web* 17 (Dec. 2012), pp. 25–32. ISSN: 15708268. DOI: 10.1016/j.websem.2012.05.003. URL: http://linkinghub.elsevier.com/retrieve/pii/S1570826812000571 (visited on 02/14/2014).

[20] Dejing Dou, Drew McDermott, and Peishen Qi. "Ontology translation on the semantic web". In: *On the Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*. Springer, 2003, pp. 952–969. URL: http://link.springer.com/chapter/10.1007/978-3-540-39964-3_60 (visited on 12/11/2015).

[21] Cynthia Dwork and Aaron Roth. "The Algorithmic Foundations of Differential Privacy". en. In: *Foundations and Trends® in Theoretical Computer Science* 9.3-4 (2013), pp. 211–407. ISSN: 1551-305X, 1551-3068. DOI: 10.1561/0400000042. URL: http://www.nowpublishers.com/articles/foundations-and-trends-in-theoretical-computer-science/TCS-042 (visited on 03/31/2019).

[22] Paul N Edwards. "Infrastructure and modernity: Force, time, and social organization in the history of sociotechnical systems". en. In: *Modernity and technology* 1 (2003), p. 35.

[23] Tolga Eren et al. "Rigidity, computation, and randomization in network localization". In: *INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE, 2004, pp. 2673–2684. (Visited on 04/15/2014).

[24] Hamza Fawzi, Paulo Tabuada, and Suhas Diggavi. "Secure Estimation and Control for Cyber-Physical Systems Under Adversarial Attacks". In: *IEEE Transactions on Automatic Control* 59.6 (June 2014), pp. 1454–1467. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2014.2303233. (Visited on 10/14/2016).

[25] Susel Fernandez et al. "Ontology-Based Architecture for Intelligent Transportation Systems Using a Traffic Sensor Network". In: *Sensors* 16.8 (2016). ISSN: 1424-8220. DOI: 10.3390/s16081287. URL: http://www.mdpi.com/1424-8220/16/8/1287.

[26] Frederico Fonseca, Clodoveu Davis, and Gilberto Câmara. "Bridging ontologies and conceptual schemas in geographic information integration". In: *GeoInformatica* 7.4 (2003), pp. 355–378. URL: http://link.springer.com/article/10.1023/A:1025573406389 (visited on 01/03/2016).

[27] Tully Foote. "tf: The transform library". en. In: *2013 IEEE Conference on Technologies for Practical Robot Applications (TePRA)*. Woburn, MA, USA: IEEE, Apr. 2013, pp. 1–6. ISBN: 978-1-4673-6225-2 978-1-4673-6223-8 978-1-4673-6224-5. DOI: 10.1109/TePRA.2013.6556373. URL: http://ieeexplore.ieee.org/document/6556373/ (visited on 05/02/2019).

[28] Andrew U. Frank. "Qualitative spatial reasoning about distances and directions in geographic space". In: *Journal of Visual Languages & Computing* 3.4 (1992), pp. 343–371. URL: http://www.sciencedirect.com/science/article/pii/1045926X92900079 (visited on 01/31/2017).

[29] R. Ghrist et al. "Topological landmark-based navigation and mapping". In: *University of Pennsylvania, Department of Mathematics, Tech. Rep* 8 (2012). URL: http://math-proxy.sas.upenn.edu/~ghrist/preprints/landmarkvisibility.pdf (visited on 07/03/2014).

[30] Nam Ky Giang et al. "Developing IoT applications in the Fog: A Distributed Dataflow approach". en. In: *2015 5th International Conference on the Internet of Things (IOT)*. Seoul, South Korea: IEEE, Oct. 2015, pp. 155–162. DOI: 10.1109/IOT.2015.7356560. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7356560 (visited on 01/28/2019).

[31] Savithri Godugula and Gregor Engels. "Survey of ontology mapping techniques". In: *Software Quality and Assurance* (2008). URL: http://www.cs.uni-paderborn.de/fileadmin/Informatik/AG-Engels/Lehre/SS08/Seminar_Software-Qualit%C3%A4tssicherung/Seminararbeiten/Vers._1.0/ontology_seminar_report_august1.pdf (visited on 12/11/2015).

[32] Steven J Gortler, Alexander D Healy, and Dylan P Thurston. "Characterizing generic global rigidity". In: *American Journal of Mathematics* 132.4 (2010), pp. 897–939.

[33] Pierre Grenon and Barry Smith. "SNAP and SPAN: Towards dynamic spatial ontology". In: *Spatial cognition and computation* 4.1 (2004), pp. 69–104. URL: http://www.tandfonline.com/doi/pdf/10.1207/s15427633scc0401_5 (visited on 01/03/2016).

[34] Munish Gupta. *Akka essentials*. Packt Publishing Ltd, 2012.

[35] Steve Harris, Andy Seaborne, and Eric Prud'hommeaux. *SPARQL 1.1 Query Language*. Mar. 2013. URL: https://www.w3.org/TR/sparql11-query/ (visited on 11/28/2018).

[36] Tian He et al. "Range-free localization schemes for large scale sensor networks". In: *Proceedings of the 9th annual international conference on Mobile computing and networking*. ACM, 2003, pp. 81–95. (Visited on 02/27/2014).

[37] Carl Hewitt, Peter Bishop, and Richard Steiger. "A Universal Modular ACTOR Formalism for Artificial Intelligence". In: *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*. IJCAI'73. Stanford, USA: Morgan Kaufmann Publishers Inc., 1973, pp. 235–245. URL: http://dl.acm.org/citation.cfm?id=1624775.1624804.

[38] Ville Honkavirta et al. "A comparative survey of WLAN location fingerprinting methods". In: *Positioning, Navigation and Communication, 2009. WPNC 2009. 6th Workshop on*. IEEE, 2009, pp. 243–251. (Visited on 03/04/2014).

[39] Todd E. Humphreys et al. "Assessing the spoofing threat: Development of a portable GPS civilian spoofer". In: *Proceedings of the ION GNSS international technical meeting of the satellite division*. Vol. 55. 2008, p. 56. URL: https://gps.mae.cornell.edu/humphreys_etal_iongnss2008.pdf (visited on 05/11/2017).

[40] Bill Jackson and Tibor Jordan. *Connected rigidity matroids and unique realizations of graphs*. Tech. rep. Budapest, Hungary: Eotvos University, Mar. 2003.

[41] Jan M. Rabaey. *The swarm at the edge of the cloud the new face of wireless*. Kyoto, 2011.

[42] Krzysztof Janowicz et al. "SOSA: A Lightweight Ontology for Sensors, Observations, Samples, and Actuators". en. In: *arXiv:1805.09979 [cs]* (May 2018). arXiv: 1805.09979. URL: http://arxiv.org/abs/1805.09979 (visited on 11/14/2018).

[43] Chadlia Jerad and Edward A. Lee. "Deterministic Timing for the Industrial Internet of Things". en. In: *2018 IEEE International Conference on Industrial Internet (ICII)*. Seattle, WA, USA: IEEE, Oct. 2018, pp. 13–22. ISBN: 978-1-5386-7771-1. DOI: 10.1109/ICII.2018.00010. URL: https://ieeexplore.ieee.org/document/8539099/ (visited on 11/28/2018).

[44] Benjamin Kempke et al. "SurePoint: Exploiting Ultra Wideband Flooding and Diversity to Provide Robust, Scalable, High-Fidelity Indoor Localization". en. In: ACM Press, 2016, pp. 137–149. ISBN: 978-1-4503-4263-6. DOI: 10.1145/2994551.2994570. URL: http://dl.acm.org/citation.cfm?doid=2994551.2994570 (visited on 04/04/2018).

[45] Hokeun Kim and Edward A. Lee. "Authentication and Authorization for the Internet of Things". en. In: *IT Professional* 19.5 (2017), pp. 27–33. ISSN: 1520-9202. DOI: 10.1109/MITP.2017.3680960. URL: http://ieeexplore.ieee.org/document/8057722/ (visited on 11/27/2018).

[46] Constantinos Kolias et al. "DDoS in the IoT: Mirai and Other Botnets". en. In: *Computer* 50.7 (2017), pp. 80–84. ISSN: 0018-9162. DOI: 10.1109/MC.2017.201. URL: http://ieeexplore.ieee.org/document/7971869/ (visited on 03/30/2019).

[47] Benjamin Kuipers. "The Spatial Semantic Hierarchy". In: *Artif. Intell.* 119.1-2 (May 2000), pp. 191–233. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(00)00017-5. URL: http://dx.doi.org/10.1016/S0004-3702(00)00017-5.

[48] Richard B. Langley. "Dilution of Precision". In: *GPS WORLD* (1999).

[49] Heiner Lasi et al. "Industry 4.0". en. In: *Business & Information Systems Engineering* 6.4 (Aug. 2014), pp. 239–242. ISSN: 1867-0202. DOI: 10.1007/s12599-014-0334-4. URL: http://link.springer.com/10.1007/s12599-014-0334-4 (visited on 03/25/2019).

[50] Elizabeth Latronico et al. "A Vision of Swarmlets". In: *Internet Computing, IEEE* 19.2 (2015), pp. 20–28. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7006378 (visited on 07/23/2015).

[51] Patrick Lazik et al. "ALPS: A Bluetooth and Ultrasound Platform for Mapping and Localization". en. In: ACM Press, 2015, pp. 73–84. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809727. URL: http://dl.acm.org/citation.cfm?doid=2809695.2809727 (visited on 04/22/2016).

[52] Daniel Le Berre. "Sat4j: a reasoning engine in Java based on the SATisfiability problem (SAT)". In: (2010).

[53] Edward Ashford Lee. *Plato and the Nerd: The Creative Partnership of Humans and Technology*. The MIT Press, 2017. ISBN: 0262036487, 9780262036481.

[54] Edward Ashford Lee and Sanjit Arunkumar Seshia. *Introduction to Embedded Systems - A Cyber Physical Systems Approach - Second Edition*. English. E. A. Lee and S. A. Seshia, July 2015. ISBN: 978-1-312-42740-2.

[55] Edward A. Lee et al. "The Swarm at the Edge of the Cloud". In: *IEEE Design & Test* 31.3 (June 2014), pp. 8–20. ISSN: 2168-2356, 2168-2364. DOI: 10.1109/MDAT.2014.2314600. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6781658 (visited on 01/27/2015).

[56] Jiyeong Lee et al. "OGC IndoorGML version 1.0". In: (2014). (Visited on 05/11/2016).

[57] Jackie Man-Kit Leung et al. "Scalable Semantic Annotation using Lattice-based Ontologies". In: *12th International Conference on Model Driven Engineering Languages and Systems (MODELS 09)*. ACM/IEEE. Oct. 2009, pp. 393–407. URL: http://chess.eecs.berkeley.edu/pubs/611.html.

[58] Ben Lickly. *Static Model Analysis with Lattice-based Ontologies*. Tech. rep. UCB/EECS-2012-212. PhD Thesis. EECS Department, University of California, Berkeley, Nov. 2012. URL: http://www.eecs.berkeley.edu/Pubs/TechRpts/2012/EECS-2012-212.html.

[59] Ben Lickly et al. "A Practical Ontology Framework for Static Model Analysis". In: *International Conference on Embedded Software (EMSOFT)*. ACM, 2011, pp. 23–32. URL: http://chess.eecs.berkeley.edu/pubs/862.html.

[60] Joshua Lieberman, Raj Singh, and Chris Goad. *W3C Geospatial Ontologies*. Tech. rep. Oct. 2007. URL: http://www.w3.org/2005/Incubator/geo/XGR-geo-ont/ (visited on 01/03/2016).

[61] Donggang Liu et al. "Attack-Resistant Location Estimation in Wireless Sensor Networks". en. In: *ACM Transactions on Information and System Security* 11.4 (July 2008), pp. 1–39. ISSN: 10949224. DOI: 10.1145/1380564.1380570. (Visited on 05/10/2016).

[62] Hui Liu et al. "Survey of Wireless Indoor Positioning Techniques and Systems". In: *IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews)* 37.6 (Nov. 2007), pp. 1067–1080. ISSN: 1094-6977. DOI: 10.1109/TSMCC.2007.905750. (Visited on 02/27/2014).

[63] Johan Löfberg. "YALMIP: A toolbox for modeling and optimization in MATLAB". In: *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*. IEEE. 2004, pp. 284–289.

[64] Marten Lohstroh and Edward A. Lee. "An interface theory for the internet of things". In: *Software Engineering and Formal Methods*. Springer, 2015, pp. 20–34. URL: http://link.springer.com/chapter/10.1007/978-3-319-22969-0%5C_2 (visited on 12/13/2016).

[65] Marten Lohstroh and Edward A Lee. "Deterministic Actors". In: *2019 Forum on Specification and Design Languages (FDL)*. IEEE. 2019.

[66] Marten Lohstroh et al. "Actors Revisited for Time-Critical Systems". en. In: *Proceedings of the 56th Annual Design Automation Conference 2019 on - DAC '19*. Las Vegas, NV, USA: ACM Press, 2019, pp. 1–4. ISBN: 978-1-4503-6725-7. DOI: 10.1145/3316781.3323469. URL: http://dl.acm.org/citation.cfm?doid=3316781.3323469 (visited on 07/14/2019).

[67] Nathan Michael, Jonathan Fink, and Vijay Kumar. "Experimental testbed for large multirobot teams". In: *Robotics & Automation Magazine, IEEE* 15.1 (2008), pp. 53–61. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4476329 (visited on 04/22/2016).

[68] Robin Milner. *The Space and Motion of Communicating Agents*. 1st. New York, NY, USA: Cambridge University Press, 2009.

[69] David Moore et al. "Robust distributed network localization with noisy range measurements". In: *Proceedings of the 2nd international conference on Embedded networked sensor systems*. ACM, 2004, pp. 50–61. URL: http://dl.acm.org/citation.cfm?id=1031502 (visited on 04/21/2014).

[70] *MOSEK related publications*. Tech. rep. Aug. 2011. URL: https://download.mosek.com/docs/reports/publications.pdf.

[71] S. Mumtaz et al. "Cognitive vehicular communication for 5G". In: *IEEE Communications Magazine* 53.7 (July 2015), pp. 109–117. ISSN: 0163-6804. DOI: 10.1109/MCOM.2015.7158273.

[72] Edwin Olson. "AprilTag: A robust and flexible visual fiducial system". en. In: *2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, May 2011, pp. 3400–3407. ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5979561. URL: http://ieeexplore.ieee.org/document/5979561/ (visited on 04/26/2019).

[73] Fabio Pasqualetti, Florian Dorfler, and Francesco Bullo. "Attack Detection and Identification in Cyber-Physical Systems". In: *IEEE Transactions on Automatic Control* 58.11 (Nov. 2013), pp. 2715–2729. ISSN: 0018-9286, 1558-2523. DOI: 10.1109/TAC.2013.2266831. (Visited on 10/14/2016).

[74] Eloi Teixeira Pereira. "Mobile Reactive Systems over Bigraphical Machines-A Programming Model and its Implementation". PhD thesis. UNIVERSITY OF CALIFORNIA, BERKELEY, 2015. URL: http://gradworks.umi.com/37/33/3733388.html (visited on 11/23/2015).

[75] Per Persson and Ola Angelsmark. "Calvin — Merging Cloud and IoT". en. In: *Procedia Computer Science* 52 (2015), pp. 210–217. ISSN: 18770509. DOI: 10.1016/j.procs.2015.05.059. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050915008595 (visited on 11/27/2018).

[76] Dennis Pfisterer et al. "SPITFIRE: toward a semantic web of things". en. In: *IEEE Communications Magazine* 49.11 (Nov. 2011), pp. 40–48. ISSN: 0163-6804. DOI: 10.1109/MCOM.2011.6069708. URL: http://ieeexplore.ieee.org/document/6069708/ (visited on 11/08/2018).

[77] Clemens Portele. "OpenGIS® Geography Markup Language (GML) Implementation Specification, version". In: (2007). (Visited on 05/03/2016).

[78] Nissanka B. Priyantha, Anit Chakraborty, and Hari Balakrishnan. "The cricket location-support system". In: *Proceedings of the 6th annual international conference on Mobile computing and networking*. 2000, pp. 32–43. URL: http://dl.acm.org/citation.cfm?id=345917 (visited on 10/31/2013).

[79] Claudius Ptolemaeus. *System design, modeling, and simulation: using Ptolemy II*. English. [Berkeley]: Ptolemy.org, 2014. ISBN: 978-1-304-42106-7 1-304-42106-6.

[80] Ira Rubinstein and Nathan Good. "Privacy by design: A counterfactual analysis of Google and Facebook privacy incidents". In: (2012). URL: http://papers.ssrn.com/sol3/papers.cfm?abstract_id=2128146 (visited on 08/30/2016).

[81] Alberto Sangiovanni-Vincentelli. "Quo Vadis, SLD? Reasoning About the Trends and Challenges of System Level Design". In: *Proceedings of the IEEE* 95.3 (Mar. 2007), pp. 467–506. ISSN: 0018-9219. DOI: 10.1109/JPROC.2006.890107. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4167779 (visited on 01/23/2015).

[82] Andreas Savvides, Chih-Chieh Han, and Mani B. Srivastava. "Dynamic Fine-Grained Localization in Ad-Hoc Wireless Sensor Networks". In: (2001).

[83] J. B. Saxe. "Embeddability of weighted graphs in k-space is strongly np-hard". In: (1979).

[84] Dongjin Seo et al. "Neural Dust: An Ultrasonic, Low Power Solution for Chronic Brain-Machine Interfaces". en. In: *arXiv:1307.2196 [physics, q-bio]* (July 2013). arXiv: 1307.2196. URL: http://arxiv.org/abs/1307.2196 (visited on 03/29/2019).

[85] Glenn Shafer. "Perspectives on the theory and practice of belief functions". en. In: *International Journal of Approximate Reasoning* 4.5-6 (Sept. 1990), pp. 323–362. ISSN: 0888613X. DOI: 10.1016/0888-613X(90)90012-Q. URL: https://linkinghub.elsevier.com/retrieve/pii/0888613X9090012Q (visited on 04/02/2019).

[86] Yasser Shoukry et al. "IMHOTEP-SMT: A Satisfiability Modulo Theory Solver For Secure State Estimation". In: *13th International Workshop on Satisfiability Modulo Theories (SMT)*. 2015. (Visited on 10/13/2016).

[87] Yasser Shoukry et al. "SMC: Satisfiability Modulo Convex Optimization". In: *Proceedings of the 10th International Conference on Hybrid Systems: Computation and Control (HSCC)*. Apr. 2017.

[88] P. Shvaiko and J. Euzenat. "Ontology Matching: State of the Art and Future Challenges". In: *IEEE Transactions on Knowledge and Data Engineering* 25.1 (Jan. 2013), pp. 158–176. ISSN: 1041-4347. DOI: 10.1109/TKDE.2011.253. URL: http://ieeexplore.ieee.org/document/6104044/ (visited on 05/31/2017).

[89] J. E. Siegel, D. C. Erb, and S. E. Sarma. "A Survey of the Connected Vehicle Landscape—Architectures, Enabling Technologies, Applications, and Development Areas". In: *IEEE Transactions on Intelligent Transportation Systems* 19.8 (Aug. 2018), pp. 2391–2406. ISSN: 1524-9050. DOI: 10.1109/TITS.2017.2749459.

[90] Anthony Man-Cho So and Yinyu Ye. "Theory of semidefinite programming for Sensor Network Localization". en. In: *Mathematical Programming* 109.2-3 (Jan. 2007), pp. 367–384. ISSN: 0025-5610, 1436-4646. DOI: 10.1007/s10107-006-0040-1. (Visited on 07/06/2016).

[91] Stefano Spaccapietra et al. "On Spatial Ontologies". In: (2004). URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.88.7653&rep=rep1&type=pdf (visited on 01/03/2016).

[92] Springer India-New Delhi. "Automotive Revolution & Perspective Towards 2030". In: *Auto Tech Review* 5.4 (Apr. 2016), pp. 20–25. ISSN: 2250-3390. DOI: 10.1365/s40112-016-1117-8. URL: https://doi.org/10.1365/s40112-016-1117-8.

[93] Stavros Tripakis et al. "A modular formal semantics for Ptolemy". en. In: *Mathematical Structures in Computer Science* 23.04 (Aug. 2013), pp. 834–881. ISSN: 0960-1295, 1469-8072. DOI: 10.1017/S0960129512000278. URL: http://www.journals.cambridge.org/abstract_S0960129512000278 (visited on 10/03/2014).

[94] Sebastian Tschiatschek et al. "Learning Mixtures of Submodular Functions for Image Collection Summarization". In: *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1*. NIPS'14. event-place: Montreal, Canada. Cambridge, MA, USA: MIT Press, 2014, pp. 1413–1421. URL: http://dl.acm.org/citation.cfm?id=2968826.2968984.

[95] Laure Vieu. "Spatial and Temporal Reasoning". In: ed. by Oliviero Stock. Dordrecht: Springer Netherlands, 1997, pp. 5–41. ISBN: 978-0-585-28322-7. URL: http://dx.doi.org/10.1007/978-0-585-28322-7_1.

[96] Zizhuo Wang et al. "Further Relaxations of the Semidefinite Programming Approach to Sensor Network Localization". en. In: *SIAM Journal on Optimization* 19.2 (Jan. 2008), pp. 655–673. ISSN: 1052-6234, 1095-7189. DOI: 10.1137/060669395. URL: http://epubs.siam.org/doi/10.1137/060669395 (visited on 02/26/2018).

[97] Roy Want et al. "The active badge location system". In: *ACM Transactions on Information Systems (TOIS)* 10.1 (1992), pp. 91–102. URL: http://dl.acm.org/citation.cfm?id=128759 (visited on 02/27/2014).

[98] Matthew Weber, Ravi Akella, and Edward A Lee. "Service Discovery for The Connected Car with Semantic Accessors". en. In: *IEEE Intelligent Vehicles Symposium* (2019), p. 6.

[99] Matthew Weber and Edward Lee. "A model for semantic localization". en. In: ACM Press, 2015, pp. 350–351. ISBN: 978-1-4503-3475-4. DOI: 10.1145/2737095.2742933. URL: http://dl.acm.org/citation.cfm?doid=2737095.2742933 (visited on 01/03/2016).

[100] Matthew Weber et al. *Gordian: Formal Reasoning Based Outlier Detection for Secure Localization*. Tech. rep. UCB/EECS-2019-1. EECS Department, University of California, Berkeley, Jan. 2019. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-1.html.

[101] Peter Wegner. "Why interaction is more powerful than algorithms". In: *Communications of the ACM* 40.5 (1997), pp. 80–91. URL: http://dl.acm.org/citation.cfm?id=253801 (visited on 04/27/2015).

[102] William Weiss and Cherie D'Mello. *Fundamentals of model theory*. Topology Atlas, 2000. URL: http://people.math.sc.edu/mcnulty/modeltheory/WeissDMello.pdf (visited on 06/17/2015).

[103] Yi-Chin Wu and Stéphane Lafortune. "Enforcement of opacity properties using insertion functions". In: *Decision and Control (CDC), 2012 IEEE 51st Annual Conference on*. IEEE, 2012, pp. 6722–6728. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6426760 (visited on 03/08/2014).

[104] Zheng Yang et al. "Beyond triangle inequality: Sifting noisy and outlier distance measurements for localization". en. In: *ACM Transactions on Sensor Networks* 9.2 (Mar. 2013), pp. 1–20. ISSN: 15504859. DOI: 10.1145/2422966.2422983. URL: http://dl.acm.org/citation.cfm?doid=2422966.2422983 (visited on 05/07/2014).

[105] Zheng Yang et al. "Detecting Outlier Measurements Based on Graph Rigidity for Wireless Sensor Network Localization". In: *IEEE Transactions on Vehicular Technology* 62.1 (Jan. 2013), pp. 374–383. ISSN: 0018-9545, 1939-9359. DOI: 10.1109/TVT.2012.2220790. URL: http://ieeexplore.ieee.org/document/6316190/ (visited on 05/11/2017).

[106] Yingpei Zeng et al. "Secure localization and location verification in wireless sensor networks: a survey". en. In: *The Journal of Supercomputing* 64.3 (June 2013), pp. 685–701. ISSN: 0920-8542, 1573-0484. DOI: 10.1007/s11227-010-0501-4. (Visited on 09/02/2016).

[107] Sheng Zhong et al. "Towards a theory of robust localization against malicious beacon nodes". In: *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE, 2008, pp. 1391–1399. URL: http://ieeexplore.ieee.org/abstract/document/4509792/ (visited on 05/11/2017).