SEMANTICS FOR THE WEB OF THINGS

VICTOR CHARPENAY

Modeling the Physical World as a Collection of Things and Reasoning with their Descriptions

A dissertation submitted to the faculty of computer science and mathematics at the university of Passau in partial fulfillment of the requirements for the degree of doctor of natural sciences

Advisor: Prof. Dr. Harald Kosch

Passau, April 2019

Victor Charpenay: *Semantics for the Web of Things,* Modeling the Physical World as a Collection of Things and Reasoning with their Descriptions, © April 2019

ABSTRACT

The main research question of this thesis is to develop a theory that would provide foundations for the development of Web of Things (WoT) systems. A theory for WoT shall provide a model of the 'things' WoT agents relate to such that these relations determine what interactions take place between these agents. This thesis presents a knowledge-based approach in which the semantics of WoT systems is given by a transformation (an homomorphism) between a graph representing agent interactions and a knowledge graph describing 'things'. It focuses on three aspects of knowledge graphs in particular: the *vocabulary* with which assertions can be made, the *rules* that can be defined over this vocabulary and its *serialization* to efficiently exchange pieces of a knowledge graph. Each aspect is developed in a dedicated chapter, with specific contributions to the state-of-the-art.

The need for a unified vocabulary to describe 'things' in WoT and the Internet of Things (IoT) has been identified early on in the literature. Many proposals have been consequently published, in the form of Web ontologies. In Ch. 2, a systematic review of these proposals is being developed, as well as a comparison with the data models of the principal IoT frameworks and protocols. The contribution of the thesis in that respect is an alignment between the Thing Description (TD) model and the Semantic Sensor Network (SSN) ontology, two standards of the World Wide Web Consortium (W3C). The scope of this thesis is generally limited to Web standards, especially those defined by the Resource Description framework (RDF).

Web ontologies do not only expose a vocabulary but also rules to extend a knowledge graph by means of reasoning. Starting from a set of TD documents, new relations between 'things' can be "discovered" this way, indicating possible interactions between the servients that relate to them. The experiments presented in Ch. 3 were done on the basis of this semantic discovery framework on two use cases: a building automation use case provided by Intel Labs and an industrial control use case developed internally at Siemens. The relations to discover often involve anonymous nodes in the knowledge graph: the chapter also introduces a novel skolemization algorithm to correctly process these nodes on a well-defined fragment of the Web Ontology Language (OWL).

Finally, because this semantic discovery framework relies on the exchange of TD documents, Ch. 4 introduces a binary format for RDF that proves efficient in serializing TD assertions such that even the smallest WoT agents, i.e. micro-controllers, can store and process them. A formalization for the semantics-preserving compaction and

querying of TD documents is also introduced in this chapter, at the basis of an embedded RDF store called the µRDF store. The ability of all WoT agents to query logical assertions about themselves and their environment, as found in TD documents, is a first step towards knowledge-based intelligent systems that can operate autonomously and dynamically in a decentralized way. The µRDF store is an attempt to illustrate the practical outcomes of the theory of WoT developed throughout this thesis.

ZUSAMMENFASSUNG

Die Dissertation entwickelt eine theoretische Grundlage für die Spezifikation *Web of Things* (WoT)-Systemen. Die Spezifikation der WoT-Systeme basiert auf einem Modell für die Dinge, oder *Things*, mit denen WoT-Agenten Beziehungen schaffen, welche Interaktionen zwischen Agenten erlauben. Diese Dissertation stellt einen wissensbasierten Ansatz vor, in dem die Semantik von Wot-Systemen als eine Transformation von einem Graphen von Agenten-Interaktionen nach einem *Knowledge Graph* definiert ist. Diese Arbeit deckt genau drei Aspekte Knowledge Graphs ab: das Vokabular, mit dem logische Schlüsse formuliert werden, die Regeln, die auf einem Vokabular basieren können und Serialisierung, um den effizienten Austausch zwischen Teilen eines Knowledge Graph zu ermöglichen. Alle drei Aspekt werden mit ihren wissenschaftlichen Beitrag in einem eigenen Kapitel adressiert.

Der Bedarf an einem vereinigten Vokabular, um im WoT und dem *Internet of Things* (IoT) Things zu beschreiben, wurden in der Literatur frühzeitig identifiziert. Viele Ansatze wurden diesbezüglich vor allem als Web Ontologien veröffentlicht. Im Kapitel 2 werden diese Ansätze miteinander, sowie mit Datenmodelle dominierender IoT-Frameworks und Protokolle verglichen. Der Beitrag der Dissertation diesbezüglich ist die Verschmelzung des WoT *Thing Description* (TD) Modells und der *Semantic Sensor Network* (SSN) Ontologie, zwei vom *World Wide Web Consortium* (W3C) veröffentlichte Standards, in eine einzige Ontologie. Der Rahmen dieser Dissertation wird auf Web Standards begrenzt, insbesondere im *Resource Description Framework* (RDF) enthaltenen Standards.

Web Ontologien bestehen nicht nur aus einm Vokabular, sondern auch aus Regel, um einen Knowledge Graphen durch Inferenz zu erweitern. Anhand einer Menge von TD-Dokumenten können neue Beziehungen zwischen Things abgeleitet werden und dadurch neue Interaktionen zwischen denjenigen Agenten, die sich auf diese Things beziehen eingeführt werden. Die im Kapitel 3 beschriebenen Experimente setzen dieses semantische Framework in zwei Domäne um: Gebäudeautomatisierung und Industrielle Kontrollsysteme. Das Erkennen impliziter Beziehungen zwischen Things hängt in bestimmten Fällen von sogenannten anonymen Knoten im Graphen ab: das Kapitel führt einen neuen *Skolemization* Algorithmus ein, um diese Knoten für einen bestimmten Teil der *Web Ontology Language* (OWL) korrekt zu verarbeiten.

Zum Schluss, da die Umsetzung dieses semantischen Frameworks den Austausch von TD-Dokumenten erfordert, wird im Kapitel 4 ein binäres Format für RDF eingeführt, welches sich als sehr effizient für die Serialisierung erweist, damit auch kleine WoT Agenten, nämlich Mikrocontroller, TD-Dokumente speichern und verarbeiten können. Eine formale Definition für die Verdichtung und die Abfrage von TD-Dokumenten wird in diesem Kapitel eingeführt. Das Kapitel beschreibt auch die Implementierung einer eingebetteten RDF Datenbank, die µRDF Store genannt wurde. Die Fähigkeit WoT-Agenten logische Schlüsse über sich selbst und ihre Umgebung zu ziehen ist der erste Schritt in Richtung eines wissensbasierten intelligenten Systems, das autonom, dynamisch und dezentral agieren kann. Der µRDF Store zeigt die praktischen Vorteile, der in dieser Dissertation entwickelten Theorie für WoT auf.

CONTENTS

1 INTRODUCTION 1
- 1.1 The Emergence of the Web of Things 1
- 1.2 Problem Statement: Semantics for the Web of Things 2
  - 1.2.1 A Brief History 3
  - 1.2.2 The Problem of Identifying & Describing 'Things' 4
  - 1.2.3 An Abstraction of Web of Things Interactions 6
- 1.3 Overview of the Thesis 8
  - 1.3.1 Outline 8
  - 1.3.2 Contributions 8
  - 1.3.3 Scope of the Thesis 10
  - 1.3.4 Methodology 10

2 MODELING 'THINGS' ON THE WEB OF THINGS 13
- 2.1 Introduction 13
- 2.2 Related Work: Ontologies for the Web of Things 14
  - 2.2.1 Introduction to the Resource Description Framework 14
  - 2.2.2 The Web of Things Ontology Cloud 17
  - 2.2.3 Other Information Models 20
- 2.3 What is a 'Thing'? 22
  - 2.3.1 Alignments Between Non-RDF Standards 23
  - 2.3.2 Alignments in the Web of Things Ontology Cloud 27
  - 2.3.3 The Thing Description Model 30
- 2.4 Note on Using Web Identifiers for 'Things' 35
- 2.5 Summary 38

3 REASONING WITH THING DESCRIPTIONS 41
- 3.1 Introduction 41
- 3.2 Related Work: Existential Reasoning 42
  - 3.2.1 RDF Graph Canonicalization 43
  - 3.2.2 Reasoning with Web ontologies 45
  - 3.2.3 Query Answering and Abduction 49
- 3.3 A Framework for Semantic Discovery on the Web of Things 52
  - 3.3.1 Expressiveness of a Web of Things Knowledge Base 52
  - 3.3.2 Problem Statement 57
  - 3.3.3 Generalization 59
  - 3.3.4 A Tractable Approach 60
- 3.4 Proof of Concept & Evaluation 64
  - 3.4.1 Use Case: Intel Labs Sensor Network 65
  - 3.4.2 Use Case: Water Management Plant 68
- 3.5 Summary 70

4 EXCHANGING THING DESCRIPTIONS 73

4.1 Introduction 73
4.2 Related Work: the Embedded Semantic Web 74
4.2.1 The Embedded Web 75
4.2.2 Serializing RDF on the Embedded Web 76
4.3 The µRDF Store: RDF on Micro-controllers 80
4.3.1 JSON-LD Expressions and Semantics 81
4.3.2 JSON-LD Frame Matching 84
4.4 Evaluation 87
4.4.1 Binary JSON-LD & Compaction 87
4.4.2 Semantic Discovery Exchanges 90
4.4.3 JSON-LD Framing Benchmark 92
4.5 Summary 95
5 CONCLUSION & PERSPECTIVES 97
5.1 Theoretical & Experimental Results 97
5.2 Future Work 98
5.3 Final Word: the Lifecycle of a Revolution 99

BIBLIOGRAPHY 102
INDEX 115
A AN OBJECT NOTATION FOR WEB ONTOLOGIES 119
B PREFIX & CONTEXT DEFINITIONS 123
C SOCIETY & THE WEB OF THINGS 127

LIST OF FIGURES

Figure 1 Water management plant model with physical tagging . . . 5
Figure 2 Evolution over time of LOV4IoT . . . 18
Figure 3 Main Concepts of WoT-related standards (UML notation) . . . 24
Figure 4 Graph of alignments for WoT-related standards 25
Figure 5 Graph of alignments for LOV4IoT . . . 28
Figure 6 WoT cloud concepts . . . 32
Figure 7 Illustration of the IRE pattern . . . 37
Figure 8 Expressiveness of logic formalisms with existential restrictions related to DLs . . . 48
Figure 9 Classes for physical bodies and their properties 55
Figure 10 Intel labs sensor network's map . . . 65
Figure 11 Intel Labs sensor network's graph of interactions 67
Figure 12 Evolution over time of coverage in the Intel Labs sensor network . . . 68
Figure 13 Overview of a water management plant model 69
Figure 14 Graph of interactions for a water management plant model . . . 71
Figure 15 Size of datasets serialized in compact binary formats . . . 90
Figure 16 Distribution of size values for sets of individual documents serialized in compact binary formats . . . 91
Figure 17 Size of ABox assertions exchanged between servients and Thing Directory . . . 92
Figure 18 Evolution of the number of intermediate results for LUBM queries on a small synthetic dataset . . . 94
Figure 19 Evolution of the CoAP RTT over the number of ABox assertions in an LUBM dataset . . . 95

LIST OF TABLES

Table 1 Component description of a water management plant model 6
Table 2 Typographical conventions 11
Table 3 Communication standards related to WoT and OWL mapping 21
Table 4 Graph statistics for WoT-related standards . . 26
Table 5 Graph statistics for LOV4IoT 29
Table 6 List of URI schemes suitable to identify ‘things’ 38
Table 7 List of URN namespaces suitable to identify ‘things’ 38
Table 8 DL features and their contribution to existential reasoning 53
Table 9 Web standards and their equivalence with Embedded Web technologies 75
Table 10 Statistics on datasets designed for compaction experiments 88
Table 11 Size of LUBM queries and answers on a small synthetic dataset 93
Table 12 Mapping of OWL to the object notation introduced in the thesis 119
Table 13 List of prefixes used in the thesis 123
Table 14 Global context for the examples in the thesis . 125

LIST OF ALGORITHMS

Figure 1 Skolemization algorithm for a knowledge base $\mathcal{K}$ with existential retrictions. S is a subset of the individual names in $\mathcal{K}$ and n an arbitrary integer. 61

Figure 2 Frame matching algorithm for a flattened graph $\mathcal{G}$ and a flattened frame $\mathcal{F}$. 86

LIST OF ACRONYMS

6LOWPAN	IPv6 Low-Power Wireless Personal Area Network
ALP	Abductive Logic Programming
BA	Building Automation
BACNET	BA Control Network
BCQ	Boolean CQ
BIM	Building Information Model
BLE	Bluetooth Low-Energy
BOT	Building Topology Ontology
BSBM	Berlin SPARQL Benchmark
CBOR	Constrained Binary Object Representation
COAP	Constrained Application Protocol
CORE	Constrained RESTful Environment
CQ	Conjunctive Query
DL	Description Logic
DLP	DL Programs
DUL	Dolce+DnS Ultralite
EDDL	Electronic Device Description Language
ELP	$\mathcal{EL}$ Programs
ERI	Efficient RDF Interchange
ETSI	European Telecommunication Standards Institute
EXI	Efficient XML Interchange
EXI4JSON	EXI for JSON
FOAF	Friend Of A Friend
FOL	First-Order Logic
GATT	(BLE) Generic Attributes
GTIN	Global Trade Identification Number
HDT	Header-Dictionary-Triples
HVAC	Heating, Ventilation & Air Conditioning
HTML	Hypertext Markup Language
HTO	Haystack Tagging Ontology
HTTP	Hypertext Transfer Protocol
IANA	International Assigned Numbers Authority
ICS	Industrial Control Systems
IETF	Internet Engineering Task Force
IFC	Industry Foundation Classes
IOT	Internet of Things
IOT-O	IoT Ontology
IP	Internet Protocol
IRE	Identity/Resource/Entity
IRI	Internationalized URI
ISBN	International Standard Book Number
JSON	JavaScript Object Notation

JSON-LD JSON for Linked Data
LDP Linked Data Platform
LOV Linked Open Vocabulary
LOV4IOT LOV for the IoT
LUBM Lehigh University Benchmark
LWM2M Light-Weight Machine-to-Machine
M3 Machine-to-Machine Measurement
MAS Multi-Agent System
MCU Micro-controller Unit
MSM Minimal Service Model
NFC Near-Field Communication
OCF Open Connectivity Foundation
OM Ontology of Units of Measure
OMA Open Mobile Alliance
OPC Open Platform Communications
OPC-UA OPC Unified Architecture
OWL Web Ontology Language
OWL-S OWL for Web Services
QU Quantity and Units
QUDT Quantity Kinds, Units and Data Types
RAM Random Access Memory
RDF Resource Description Framework
RDF4LED RDF for Lightweight Edge Devices
RFID Radio-Frequency identification
RTT Round-Trip Time
SAN Semantic Actuator Network
SAREF Smart Appliance Reference (model)
SEAS Smart Energy Aware System
SKOS Simple Knowledge Organization System
SOSA Sensor, Observation, Sample and Actuator
SPARQL SPARQL Protocol and RDF Query Language
SSN Semantic Sensor Network
SWEET Semantic Web Earth and Environment Terminology
TD Thing Description
TDIR Thing Directory
UCUM Unified Code for Units of Measure
UML Unified Modeling Language
URI Uniform Resource Identifier
URN Uniform Resource Namespace
W3C World Wide Web Consortium
WOT Web of Things
WSMO Web Service Modeling Ontology
XML Extended Markup Language

1 INTRODUCTION

1.1 THE EMERGENCE OF THE WEB OF THINGS

The idea of a Web of 'things' that emerged in the past decade has a simple premise: there is more and more interconnected computing agents involved in industrial networks. Making these agents interact requires a high level of interoperability, which requires in turn to rethink the general architecture of industrial networks. Web technologies, which have always been designed for system interoperability, represent a good approach to solve this issue. The present work is an attempt to formalize agent interactions in this context and develop a theory to make interactions be carried out in an autonomous way. Before addressing the core of the topic, let us relate an anecdote to understand what motivated such an approach and why interoperability has become paramount in industrial systems.

In 2010, a malicious computer program later named "Stuxnet" was detected on millions of industrial controllers around the world [41]. This program was designed to scan a network to find Siemens Simatic S7 controllers that connected to field devices over a Profibus industrial bus. Profibus devices have a 16-bit identifier; Stuxnet was looking specifically for devices with the identifier `0x9500`. Moreover, the program was expecting a structure of 6×64 field devices connected to the same S7 instance. After several years of investigation, analysts inferred from these elements that Stuxnet was targeting a uranium enrichmnent plant in Natanz, Iran. To come to this conclusion, they also had to gather background knowledge about industrial networks, in particular the fact that `0x9500` is the identifier used by Siemens for frequency converters, that this kind of devices is usually used to control centrifuges and that the structure coded in Stuxnet corresponded to the way centrifuges were organized in the Natanz facility.

The discovery of Stuxnet sheds light on two aspects of industrial networks. First, the extent to which Stuxnet spread across the world indicates how far the Internet has reached specialized networks like those used in plants and large buildings. Indeed, most industrial controllers, like Siemens S7, have an interface to an Internet Protocol (IP) network in addition to their interface to a field bus, which is the channel Stuxnet used to spread. A quick search for public IP devices reports $4,858$ distinct IP addresses in more than 50 different countries for the keyword s7[1]. Most industrial networks are isolated from

1 request made on `https://censys.io/` on January 13th, 2019.

public Internet and this figure only represents the tip of the iceberg, which must reach millions of connected devices.

Second, besides having low security standards, it is typical of industrial networks to introduce vendor-specific or hard-coded elements at the interface between systems. It is e.g. only known to Siemens customer that `0x9500` is the identifier used for frequency converters over Profibus. Similarly, the topology of the Natanz facility with six rows of 64 centrifuges was hard-coded (devices had fixed identifiers in the network), which allowed a malicious program to discriminate this network over thousands of others.

These two trends in industrial networks are somewhat contradictory. The attempt to normalize communication in industrial systems by relying on IP protocols is hindered by the lack of standardization across systems at the application level, which translates into many hard-coded elements. Higher-level communication protocols are diverse, they do not share data models and are often domain-specific (*de facto* or by design), causing system integration to be labor-intensive and specialized. In other words, interoperability between industrial systems is low, despite a generally high connectivity.

The most successful approach towards interoperability in the past decade has consisted in providing a Web interface to individual systems, to take advantage of mature Web technologies and the general expertise among developers to build applications on top of the Web. This approach, which would extend the coverage of the Web to industrial systems is referred to in the scientific literature as the Web of Things (WoT) and the World Wide Web Consortium (W3C) is working on standardizing some of its building blocks. In fact, most industrial controllers expose a back-end Web server for remote administration, which makes them theoretically part of WoT already.

Contrary to current industrial systems, whose hierarchical design addresses strict functional requirements like low latency and deterministic behaviors, the Web is highly decentralized. WoT systems therefore imply complex hyperlink-driven interactions that must be formalized if these systems are to meet similar requirements. The set of WoT technologies that the W3C is gathering addresses numerous aspects of WoT but it leaves room for a theory of WoT systems and their interactions. This thesis presents a knowledge-based approach to this problem by relying on Web ontologies and their formal semantics.

1.2 PROBLEM STATEMENT: SEMANTICS FOR THE WEB OF THINGS

Starting from a history of WoT and its terminology, it is possible to choose various focus points to aim at in a scientific research work. The brief history exposed in the following leads to the problem of identifying and describing ‘things’ in systems made of sensors and actuators, which is introduced in a dedicated section. Then, a theory of WoT systems based on graph structures is being developed.

1.2.1 *A Brief History*

The term 'Web of Things' echoes the earlier concept of Internet of Things (IoT) [1, 46]. The IoT itself greatly overlaps with research fields that predate it and that developed their own terminology. We start by summarizing the main fields from which the present work work and most theoretical works on WoT borrow, in chronological order:

CYBERNETICS (1948) Science related to the automatic control of systems in a broad sense, from industrial systems to living organisms [121]. One of the central concepts of cybernetics is that of feedback loop, a structure in which the controlling part of a system can observe the effects of actions of its mechanical part.

UBIQUITOUS COMPUTING (1993) Method of computation including all possible forms of processors, from powerful Cloud servers to micro-controller devices [94, 120]. Ubiquitous computing aims at making computing transparent to end users of computer-based systems.

AMBIENT INTELLIGENCE (1998) Characteristic of computer systems capable of sensing their environment and adapting their behavior accordingly [122]. The term was first coined in the context of consumer electronics, it referred to the increasing capacity of electronic devices to perform general purpose computation. Ambient intelligence later converged with ubiquitous computing as one research area.

WIRELESS SENSOR NETWORKS (2000) Type of communication network designed for monitoring, usually composed of constrained nodes with sensing and routing capabilities [84]. Sensor networks have been a recurrent research field over more than 50 years but their large adoption began with new hardware design and the development of tailored operating systems like TinyOS [80] and Contiki [30].

INTERNET OF THINGS (2000) Interconnection of any digitally identified objects in the physical world with the rest of the Internet. The development of the IoT goes hand in hand with Radio-Frequency identification (RFID) [1, 103]. It shares many characteristics with Ambient Intelligence, with the exception that the IoT concentrates on the identificiation of inanimate products, e.g. for supply chain management, as opposed to electronic products that become "intelligent" by themselves.

PERVASIVE COMPUTING (2001) Synonym for Ubiquitous Computing [104]. The term 'pervasive' stresses the importance of sensing in ubiquitous computing.

CYBER-PHYSICAL SYSTEM (2007) Industrial system associating physical equipment with digital control [76]. The study of cyber-physical systems tends to subsume cybernetics by incorporating advances in embedded system research, including in the fields mentioned above.

These different definitions refer to systems of different kinds, at different scales. In particular, sensor networks only involve low-power devices communicating in a mesh network while cyber-physical systems can include complex machinery that integrates both sensing and actuation capabilities. IoT systems can even deal with inanimate objects with passive RFID tags. As a result, components of IoT (and WoT) systems are referred to as 'things', arguably the most generic term to refer to physical objects. However, although all systems considered here include sensing, the concept of 'thing' mixes the subject of sensing, i.e. a computational device, and the sensed object, like manufactured products, living beings or scenes. In Sec. 2.3, a more precise definition of 'thing' is provided, in which the distinction can be made.

The modeling of sensing and actuating devices has been extensively studied in the fields of ubiquitous computing and ambient intelligence. In this thesis, it is argued that the novelty of IoT and WoT systems lies in their capacity of exchanging information about the objects of sensing, i.e. physical world entities that have no computational capability. In contrast to modeling computational devices, modeling physical world entities in the context of industrial systems is still a research question. In particular, the problem of digitally *identifying* entities and differentiating them from each other, i.e. the problem of seeing the physical world as a collection of 'things', is technologically unsolved.

1.2.2 *The Problem of Identifying & Describing 'Things'*

Recent technologies like RFID, Bluetooth Low-Energy (BLE), Zigbee and novel long-range radio protocols like LoRa allow computational systems to collect large amounts of data about physical world entities. However, these technologies mostly operate at the transport layer. They are used to transport data of any nature, no assumption is made about the content of exchanged messages. For instance, an RFID tag can encode an arbitrary identifier for the tagged object in a byte array of fixed size but the protocol provides in itself no guarantee of existence or uniqueness of the encoded identifier.

Formally, identification is a mapping of physical world entities to a countable space like the set of natural numbers or the set of UTF-8 strings. It is always possible to define a naive numeric identification mechanism for 'things', like assigning a serial number to manufactured products. With the example of Stuxnet, one can also see that only few features are needed to discriminate one industrial system over more than hundreds of thousands: a product type (`0x9500`) and a data structure (6×64). However, the set of 'things' to identify is potentially infinite and even if we define atomic 'things' as manufactured products and spare parts, it is likely that not every single one of them will be equipped with an RFID, BLE or Zigbee tag. The main research question of this thesis is thus to find an identification mechanism for a collection of 'things' such that a WoT system can perform various automation tasks on these 'things'. The method being develop in this work is a method to *describe* 'things' by their properties, as in the following example.

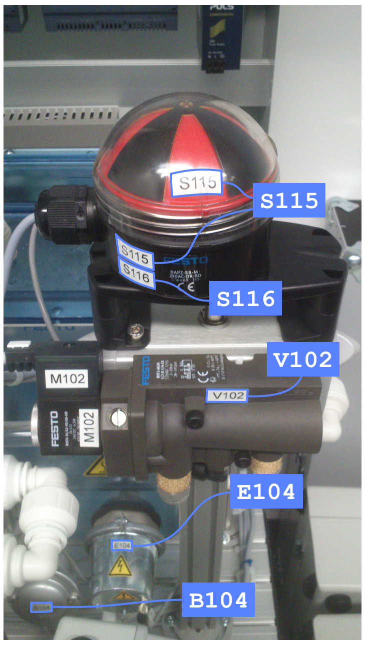


Figure 1: Close view of (parts of) a water management plant model with physical tagging

Figure 1 shows some of the components of a water management plant model used for later experiments. Various components were tagged, as highlighted on the picture. One cannot infer from the labels E104 and B104 that the corresponding devices are respectively a heater and a temperature sensor. The labels are also independent from contextual information like the fact both devices are mounted on the same object (a water tank, visible in the background). Contextual information is often relevant when describing the whole system and it can also be a substitute for abstract identifiers: B104 is equivalent to "the temperature sensor mounted on a water tank" for this particular system. Combining both methods adds redundancy and lowers the risk of misidentification of physical world entities. In Sec. 2.2.1, this mixed representation is formalized by a notation that associates a class and properties to some identifier, as in object-oriented

DEVICE	DESCRIPTION
S115	Valve opening control for V102
S116	Valve closing control for V102
V102	Pneumatic valve with digital control to empty the system's water tank
E104	Heater mounted on the system's water tank
B104	Temperature sensor mounted on the system's water tank

Table 1: Description of the physically tagged components of the water management plant model of Fig. 1

programming. The thesis includes numerous examples of such "objects" throughout this thesis. An object is the shortest formal description one can make of a physical world entity and it often easily translates into natural language. Table 1 includes textual descriptions for E104, B104, as well as S115, S116 and V102 which are parts of the same pneumatic valve. We will come back to this water management system later in Ch. 3 (Sec. 3.4.2).

The W3C has developed a Thing Description (TD) model that does not only include *properties* for a 'thing' but also the *actions* that can be performed on it and the *events* that relate to it [61]. This model was mostly developed on an empirical basis and in relation to concrete implementations. This thesis develops the theoretical foundations of the TD model in order to capture the complexity of large WoT systems. These foundations themelves rely on the concept of *knowledge graph*, in which physical world entities are described by their relations to each other in a graph. Knowledge graphs find applications on the Semantic Web, which is built on top of the Resource Description Framework (RDF). RDF is both a data model for Web resources and a set of W3C standards that includes the SPARQL Protocol and RDF Query Language (SPARQL) and the Web Ontology Language (OWL), which we explore in more details in Ch. 2 & 3. We also revisit SPARQL in Ch. 4.

1.2.3 *An Abstraction of Web of Things Interactions*

As mentioned before, the primary purpose of describing 'things' in a WoT system is to drive automation in that system. In the water management plant example, it seems intuitive that E104 and B104 be coordinated to implement e.g. a thermostat, because they relate to the temperature of the same water tank. The basic assumption of WoT is that there exist "proxies" for these two 'things' that have have computational power and that can interact over IP. It is not required that every 'thing' has its own proxy, a single chip can be a proxy for different 'things'. The general idea developed in this thesis is that an interaction between computational agents is possible if there is a relation between the 'things' for which they are a proxy, like E104 and B104. If B104 were measuring the temperature of a room, it could interact with any sensor located in the same room or in some adjacent space.

As prerequisite to formalizing this idea, a definition of WoT systems is first introduced, as well as a simple model for the interactions that take place between their components. This definition is based on that of the Web. The Web can be simply defined as a browsable collection of interlinked documents [60]. It is designed to be browsed by *user agents*, mostly humans, via a standard software stack (a browser). WoT represents a subset of the Web, in which documents provide sensor measurements and commands and in which most agents are not humans but computers. Computer agents in WoT can be considered as *autonomous*, as they are not controlled individually by humans, and are therefore "intelligent" agents. From this perspective, WoT systems are a particular kind of Multi-Agent System (MAS) that uses the architecture of the Web. We can define WoT systems as follows:

WEB OF THINGS SYSTEM Sensing and actuation system composed of autonomous and self-describing agents exposing affordances to possible interactions. The fulfilment of a given task by a WoT system results from actual interactions between these agents.

A WoT system is first described by the interactions that take place between WoT agents. An interaction can have several forms: it can be a request sent by a client to a server followed by a response, it can be a server-sent message to a client that had subscribed to certain events on the server or it could even involve a message broker for asynchronous delivery. Since agents can alternatively act as clients and servers, the generic term of 'servient' is used to describe WoT agents. An abstract model for servient interactions can be the graph $G = \langle V, E \rangle$ such that V is a set of servients and E a set of pairs $\{x, y\}$ such that servient x is involved in an interaction with servient y. Because client and server roles are mitigated in WoT, the edges of G are unordered sets (as opposed to vectors which preserve order). We refer to G as a 'graph of interactions'. This abstraction will be developed with examples in Ch. 3 (Sec. 3.4).

It is argued here that this simple graph model for WoT interactions captures most of the complexity of WoT systems. On this model, one can then base a theory that would provide the foundations for future development of WoT systems. The thesis exposes one such theory, in which the semantics of a WoT system is given by a transformation (a homomorphism) between its graph of interactions and a knowledge graph describing the 'things' it observes and acts upon.

1.3 OVERVIEW OF THE THESIS

The reader will find in the following the necessary tools to get started on reading the thesis: an outline, a summary of the scientific contributions of the thesis, its scope (defined by excluding some aspects of WoT systems that are orthogonal to the present problem statement) and, finally, the general methodology of the thesis.

1.3.1 *Outline*

This thesis focuses on constructing a knowledge graph for a WoT system, in order to analyze its graph of interactions and, by that, its semantics. In particular, three aspects of knowledge graphs are being considered: the *vocabulary* they use as defined in Web ontologies, the *rules* that can be defined to construct them and their *serialization*, such that fragments of knowledge graphs can be efficiently exchanged among WoT servients. Each aspect is developed in a dedicated chapter.

In Ch. 2, existing models for IoT and WoT systems are being reviewed, in order to formally define in RDF and OWL terms the W3C TD model. In this review, we are particularly interested in defining the concept of 'thing' according to existing Web ontologies. In Ch. 3, the semantics of Web ontologies is being introduced as sets of rules, from which assertions about 'things' can be inferred using state-of-the-art reasoning techniques. Finally, in Ch. 4, a compact binary serialization for TD documents is being introduced, suitable for WoT agents with constrained resources (e.g. memory, battery consumption). The last chapter, Ch. 5, includes a summary of all experiments presented in the thesis and a discussion of the foundational hypothesis that the semantics of interactions on WoT can be expressed with knowledge graphs.

1.3.2 *Contributions*

The main scientific contribution of this thesis is the formalization developed in Sec. 3.3, in which a graph of interactions G is labeled with a conjunctive query formulated using a WoT-specific vocabulary. In addition, every chapter includes more specific contributions related to RDF: Ch. 2 introduces a Web ontology (in the OWL format) for the W3C TD model, Ch. 3 introduces a query answering method for certain OWL ontologies and Ch. 4 formalizes the semantics of a recent object notation for RDF, from which a straightforward compaction method can be derived. Briefly:

1. The OWL formalization of the TD model mainly consists in a semantic alignment of the concepts of 'thing' defined by the TD

model to concepts from the Semantic Sensor Network (SSN) ontology: 'system', 'platform' and 'feature of interest'. The reader can already refer to Fig. 6 for a visualization of that alignment. Moreover, a systematic review of IoT and WoT models and their alignments suggests that this TD ontology is the closest semantically to all existing models. In this review, the semantic "distance" between models is computed using graph metrics (Secs. 2.3.1 & 2.3.2).

2. A particularity of WoT knowledge graphs is that they contain many "anonymous" nodes, that is, entities with no identifier but whose existence can be indirectly inferred. There has been little research so far on processing conjunctive queries on such nodes. The thesis focuses on a fragment of OWL for which a transformation procedure with polynomial complexity exists to remove anonymous nodes (what is commonly referred to as *skolemization*). Proofs for the correctness and complexity of the procedure are given in Sec. 3.3.4. Some of the theoretical details of the rule-based formalism introduced in this thesis are provided in Appendix A.
3. The state-of-the-art in compact RDF serialization roughly consists in indexing text-based identifiers and then storing indexed values in bitmaps, efficient for large amounts of data. However, this range of techniques is not directly applicable to resource-constrained servients which can only store few RDF statements. This thesis introduces a natural serialization for RDF with *contextualization* which is a formalization of the recent JSON for Linked Data (JSON-LD) format. Among others, JSON-LD includes a mechansim to compact and expand RDF documents based on a globally defined context. Unlike the state-of-the-art, there is no need to store the index (i.e. the context) along with the indexed data. In Sec. 4.4, it is showed that JSON-LD compaction combined with binary JSON serialization performs better than the state-of-the-art for TD documents.

Besides scientific contributions, two implementations of the tools used for WoT experiments have been released with open-source licenses. Both are primarily maintained by the author of this thesis, myself. The first program is a Thing Directory (TDir), a discovery agent that indexes TD documents and performs various reasoning tasks on the resulting knowledge graph, like inferring a graph of interactions[2]. The second program is a JavaScript implementation of an RDF store designed for micro-controllers[3] (µRDF.js). All datasets and experimental results of the thesis have also been published online[4].

2 https://github.com/thingweb/thingweb-directory/
3 https://github.com/vcharpenay/uRDF.js
4 https://github.com/vcharpenay/urdf-store-exp

1.3.3 *Scope of the Thesis*

Defining formal semantics for the interactions that take place in a WoT system has many potential applications, which would each require a thorough evaluation. In the present work, evaluation has been deliberately restricted to certain aspects. If we consider a WoT system roughly goes through the four phases of (1) engineering, (2) deployment, (3) production and (4) decommissioning, we can then say that the scope of the thesis was narrowed to the deployment phase. Indeed, it refers to the graph of interactions of WoT systems only for the *discovery* of potential interactions between servients after they are deployed in a particular environment.

This formalism would however also have applications in the engineering phase, e.g. as a way to estimate the coverage in terms of sensing and actuation equipment that is required to perform certain tasks. Similarly, in the production phase, graphs of interactions can help monitor exchanges in WoT systems, detect failure and dynamically adapt by updating the knowledge individual servients have of their environment (and thus allowing new interactions to take place). Conversely, other works in the context of WoT focus on the engineering and production phases and not on deployment or discovery. It is the case for a range of experiments on "recipes" for WoT applications [116] as well as works on applying the principles of Linked Data in WoT systems [5, 72]. These works were conducted in parallel to my own and are likely to be complementary rather than competing.

Moreover, the scope of this thesis is limited to approaches covered by technological standards. In the first place, the definitions that were adopted are those given by the W3C in its standardization activity for WoT. One can argue that because of the decentralized nature of the Web, the standardization of WoT building blocks is necessary if its general principles are to be implemented in practice. Knowledge graphs and ontologies are also referred to in their RDF variant and certain logical modalities like time were deliberately left aside, as they are not covered by RDF.

1.3.4 *Methodology*

The limitation in scope to standardized approaches generally influenced the methodology applied to this work. Because standard-based technologies were readily available, this thesis mostly consists in a thorough review of existing works and their applications, evaluated according to their level of maturity to be integrated into engineering processes. Another motivation was also to be able to quickly provide prototypical implementations to illustrate the benefits of WoT in an industrial environment. As a consequence, the theory of WoT being developed here is the *result* of several years of prototyping as opposed

EXAMPLE	DESCRIPTION
‘temperature’	Generic concept defined in natural language
Temperature	Well-defined concept available in the literature
`Temperature`	Term with mapping to an IRI for a given context (see Appendix B)
`om:Temperature`	CURIE (see Sec. 2.2.1)
`http://www.wurvoc.org/...`	IRI

Table 2: Typographical conventions used in this thesis to refer to ontological concepts

to being the starting point of experimental design. Parts of this model, like the TD ontology, are to be included in the W3C WoT framework[5]. In this sense, the general methodology of this thesis differs from the classical hypothetico-deductive approach of natural sciences.

Nonetheless, this work follows a strict mathematical formalism, borrowed to the most part from the literature, which finds its foundations in set theory and model theory. In addition to the mathematical notation that was adopted for formal logical expressions, I have striven to follow typographic conventions to distinguish concepts expressed in natural language from their formal counterparts. Such a distinction must indeed be made when referring to Web ontologies as not everything can be strictly formalized in this domain, at the intersection between philosophy, linguistics and computer science. The conventions I have chosen are summarized in Table 2.

5 `http://www.w3.org/ns/td` (subject to changes)

2 MODELING 'THINGS' ON THE WEB OF THINGS

2.1 INTRODUCTION

Philosophical writings have addressed the question of *individuation* long before computers even existed. From Aristotle in ancient Greece to Thomas Aquinas in the 13th century to modern philosophers, the formulation of this problem has evolved continuously as new theories of metaphysics emerge and replace old ones. Individuation can be seen as the intellectual process of naming parts of reality as objects of human perception [36]. If we translate the problem to the digital space and to machine perception, it remains essentially the same: how to model the 'things' that are perceived by digital sensors?

At the intersection between philosophy and computer science is the notion of *ontology*, which is commonly defined as a set of concepts and the logical relations between them, both defined in a way that their definition can be exchanged between computational agents [45]. The computer scientist Alonzo Church introduced a formalization for ontologies that distinguishes between an *object*, the *name* one gives to this object and the associated *concept* [27]. This formalization gave birth to object-oriented programming where concepts become classes and names become (scoped) identifiers or symbols.

In that sense, every class diagram following the Unified Modeling Language (UML) is an ontology. In the context of the Web, the dedicated formalism is OWL, a language that has its roots in computational logic and that goes much beyond UML in terms of expressiveness (although it remains within the paradigm of object-oriented modeling). OWL provides one with a first definition for the concept of 'thing' in WoT. Indeed, every object (or individual, as the result of individuation) is an instance of the class `owl:Thing`. However, when Kevin Ashton first spoke of an Internet of 'things', he had a different definition in mind. He was indeed only referring to objects that could be "tagged" with an RFID chip, namely physical or *tangible* objects [1]. As mentioned in Sec. 1.2.1, related fields of research focus on sensing and actuating devices, which also are potential 'things' but this definition still excludes intangible entities like events or human organizations, which are yet identifiable and therefore are `owl:Things`.

Starting from this observation, the first step towards defining the semantics of WoT is to formally define the concept of 'thing' in the context of WoT. In other words, we must define a (Web) ontology for WoT. This chapter presents the TD model, which is an attempt to summarize existing works addressing this problem. We start with a review of these works after a short introduction to OWL and RDF, its underlying data model (Sec. 2.2). Since not all of these works provide OWL-based alignments, a comparison is provided by analyzing their respective lexicon (Sec. 2.3). This comparison allows one to formally define the TD model and provide examples of WoT 'things' (Sec. 2.3.3). An immediate consequence of this modeling is the choice of identifiers for 'things', for which brief guidelines are provided (Sec. 2.4).

2.2 RELATED WORK: ONTOLOGIES FOR THE WEB OF THINGS

Web ontologies can have different levels of complexity, usually proportional to the engineering effort required to design them. The lowest complexity level is to define a *vocabulary*, i.e. a set of names designating concepts (the term 'signature' is also used in the literature). Every OWL concept name is also an RDF *resource*. In the following, the principles of RDF are briefly introduced and a definition for the notion of resource is given, which will suffice to understand the remainder of the chapter. Later in the thesis, we will need formal definitions for OWL to address more advanced aspects of Web ontologies (Ch. 3).

2.2.1 *Introduction to the Resource Description Framework*

The architecture of the Web has two foundations: resources and hyperlinks, respectively the vertices and edges of a world-wide corpus of documents [60]. Every resource on the Web is globally addressed by an arbitrary identifier, a Uniform Resource Identifier (URI). The specification of the Internationalized URI (IRI), which extends the character set of identifiers to UTF-8, supersedes that of URI. We will therefore only consider IRIs in this thesis.

Besides uniquely identifying a Web resource, an IRI also contains all the necessary information to request a representation of that resource using well-known protocols. Together, resource IRIs and hyperlinks allow (user) agents to browse the Web as if it were a single document. The original goal of RDF is to make hyperlink relations themselves browsable, i.e. to identify them with IRIs as well. The motivation for RDF is to let a "Semantic" Web emerge so that intelligent agents can dynamically discover the semantics of links and autonomously decide whether to follow them or not, e.g. to improve page indexing or perform advanced question answering [55]. Clearly, WoT provides a new use case for RDF as most WoT agents will browse the Web without human assistance.

The core RDF data model is based on the notion of *triples*, which are to be understood as logical statements (or assertions) composed of a *subject*, a *predicate* and an *object* [48, 54]. Because RDF is itself part of the Web, all three components can be IRIs. This way, RDF triples can be seen as conceptual links between Web resources where the link relation type is itself described by a Web resource. The object of a triple can also be a *literal* (e.g. a number or a plain string). A set of RDF triples is called an *RDF graph*, from which the concept of knowledge graph is derived. In the present writing, the two terms will be used interchangeably.

Example 1. *The following RDF graph (composed of two triples) describes a resource of type 'sensor' with link to a logo for integration in graphical user interfaces:*

s_1	`http://example.org/sensor`
p_1	`http://www.w3.org/1999/02/22-rdf-syntax-ns#type`
o_1	`http://www.w3.org/ns/sosa/Sensor`
s_2	`http://example.org/sensor`
p_2	`http://schema.org/logo`
o_2	`https://en.wikipedia.org/wiki/File:Siemens_AG_logo.svg`

Every IRI in Ex. 1 can be dereferenced to obtain a representation of the corresponding resource. For instance, o_2 would return a vector graphic of the Siemens logo. p_1, p_2 and o_1 are "semantic" resources, that is, they are part of the vocabulary of some Web ontology. p_1 is defined by the RDF Schema, the core ontology for RDF; p_2 is defined by schema.org, an extensive ontology for anything that may be exposed on the Web and referenced by search engines; o_1 is defined by the Sensor, Observation, Sample and Actuator (SOSA) ontology. For the sake of convenience, each vocabulary is exposed on the Web under its own namespace. It is a common practice to express an ontological name as a contraction of a namespace prefix (e.g. `rdf:`, `sosa:`, `schema:`) and a local name (a human-readable name). For instance, `http://www.w3.org/ns/sosa/Sensor` would become `sosa:Sensor`. This compacted form of IRI is called a CURIE.

Web ontology vocabularies incude names of three different kinds: class names, property names and individual names. The RDF convention to distinguish them is that class names are capitalized, like `sosa:Sensor`, while property names and individual names are not. The latter two can usually be distinguished by their position in a triple: predicates are always property names, while individual names shall only be used as subject or object. Class names should always appear as objects in a triple that has `rdf:type` as a predicate. It is also common practice that property names start with a verb, like `sosa:hasSubSystem`, to avoid ambiguity. The two property names in the above example (`rdf:type` and `schema:logo`) do not follow this convention, though.

For the sake of conciseness, RDF terms will only be referred to by their local name in most examples of this thesis. A compact syntax is also used to represent triples to increase readability, with two main changes compared to the idiomatic RDF representation. First, triples with the same subject are factorized. It is indeed common that the subject of a sentence relates to more than one object, in natural language as well as in formal assertions. It is the case in Ex. 1. Second, since the `rdf:type` predicate is very common in RDF graphs, syntactic sugar can be provided to shorten type statements. In its general form, an RDF graph is a set of expressions, each with a single subject, a class separated by a colon (:) and a property map within brackets ([. . .]).

Example 2. *The following expression is a syntactic variant of Ex. 1:*

$$\textit{sensor:Sensor[logo} \rightarrow \textit{Siemens\_AG\_logo.svg]}.$$

This notation can be seen as an abstraction of different serialization formats for RDF (among others, JSON-LD). We will come back to it in details in Ch. 4. It is clearly inspired by object-oriented programming but it is best to avoid mixing terminologies because of too many polysemic terms (on the term 'object' in the first place). The remainder of the thesis mostly refers to classes, properties and individuals and the RDF terminology of subject, predicate, object is avoided as far as possible. The individuals to which properties map are simply called *values*.

A Web ontology is a set of rules defined for a given vocabulary. When an RDF graph uses a vocabulary to express logical statements on arbitrary RDF resources, it implicitly "imports" these rules and the sum of the two (statements and rules) form what is called a *knowledge base*. The principles of reasoning with knowledge bases will be developed in the next chapter but for now, we quickly fast-forward through those definition necessary to introduce the concept of *ontology alignment*, the govering principle of the present chapter.

We denote N^C, N^P and N^I the disjoint sets of IRIs corresponding to class names, property names and individual names, respectively. The vocabulary defined by a knowledge base $\mathcal{K}$ is denoted $\mathsf{N}_\mathcal{K}$ and we have $\mathsf{N}_\mathcal{K} \subset \mathsf{N}^\mathsf{C} \cup \mathsf{N}^\mathsf{P} \cup \mathsf{N}^\mathsf{I}$. It is possible to say that a Web ontology is a particular kind of knowledge base whose vocabulary is fixed. To determine whether a knowledge base $\mathcal{K}$ is logically consistent, one approach consists in proving that an *interpretation* exists for $\mathcal{K}$. In this context, an interpretation $\mathcal{I}$ is roughly a function that maps elements of $\mathsf{N}_\mathcal{K}$ to parts of some graph: class names to sets of vertices, property names to sets of edges and individual names to single nodes. An interpretation $\mathcal{I}$ *satisfies* $\mathcal{K}$ if it satisfies certain constraints on the rules defined in $\mathcal{K}$.

On this basis, semantic alignment between two ontologies $\mathcal{K}$ and $\mathcal{K}'$ can be defined as the existence of some overlap in their interpretation.

Definition 1. *Let $\mathcal{K}$, $\mathcal{K}'$ be ontologies. We say that $\mathcal{K}$ semantically aligns with $\mathcal{K}'$ if one of the following holds for all $\mathcal{I}$ satisfying $\mathcal{K} \cup \mathcal{K}'$:*

- *there exists $a \in \mathsf{N}_{\mathcal{K}} \cap N^{\mathrm{I}}$ and $C \in \mathsf{N}_{\mathcal{K}'} \cap N^{\mathrm{C}}$ such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$*
- *there exists $C \in \mathsf{N}_{\mathcal{K}} \cap N^{\mathrm{C}}$ and $C' \in \mathsf{N}_{\mathcal{K}'} \cap N^{\mathrm{C}}$ such that $C^{\mathcal{I}} \subseteq C'^{\mathcal{I}}$*
- *there exists $p \in \mathsf{N}_{\mathcal{K}} \cap N^{\mathrm{P}}$ and $p' \in \mathsf{N}_{\mathcal{K}'} \cap N^{\mathrm{P}}$ such that $p^{\mathcal{I}} \subseteq p'^{\mathcal{I}}$*

Intuitively, an ontology for WoT would align to every other OWL model $\mathcal{K}$ related to sensing and actuation by defining the abstract class name Thing such that $\mathsf{C}^{\mathcal{I}} \subseteq \mathsf{Thing}^{\mathcal{I}}$ as per Def. 1 for every class name $\mathsf{C} \in \mathsf{N}_{\mathcal{K}}$ denoting either the subject or the object of sensing and actuation. The remainder of this chapter will be dedicated to the review of such ontologies (like SOSA or schema.org) in order to then define Thing in more concrete terms.

In practice, most OWL ontologies available online include axioms like `owl:subClassOf` to express alignment between terms, possibly pointing at different vocabularies. This results in an "ontology cloud" of interlinked OWL documents. This cloud is e.g. materialized on the Linked Open Vocabulary (LOV) platform, a catalogue maintained by a community [117]. We review next the fragment of the LOV cloud that relates to WoT and the IoT.

2.2.2 *The Web of Things Ontology Cloud*

The LOV platform is not an exhaustive source, ontology documents must follow certain guidelines to be published on LOV while a significant part of the potential contributions to the state-of-the-art does not follow these guidelines, for lack of time and technical expertise. To bridge this gap, Gyrard et al. maintain a catalogue of IoT ontologies and domain-specific vocabularies relevant for the IoT that are potential but incomplete contributions to LOV [49]. This catalogue, LOV for the IoT (LOV4IoT), included 462 contributions as of September 2018[1]. "Contributions" is to be understood in a broad sense here: any formal model expressed in RDF and mentioned in a scientific publication is a potential contribution, even those that are not formalized in OWL or are not available on the Web as dereferenceable resources.

The vocabularies of LOV4IoT are classified in different categories depending on the domain they address. There are vocabularies for e.g. food & beverages, healthcare, environmental monitoring, agriculture or transport. Such ontologies do not necessarily include communication aspects or a model for computational agents. They model a

1 see `https://lov4iot.appspot.com/?p=ontologies` for the latest status.

certain domain of application without considering the internal mechanisms of the system applied in that particular domain. An application integrates several domain-specific ontologies that must be aligned with each other, which is typically done by aligning them to the same *upper* ontology, an abstract model analogous to meta-models in other formalisms like UML. Upper ontologies are those of interest in the present attempt to define in OWL the abstract class Thing. In LOV4IoT, upper ontologies with system description are classified in the categories 'WoT' and 'IoT'. For these two categories, there are 58 contributions available.

Later in this chapter, we will look at several LOV4IoT contributions individually. At a macro-level, an analysis of the history of LOV4IoT suggests that the research community is currently moving away from the question of ontology engineering for WoT, as shown on Fig. 2. 21% of the contributions were not dated and therefore are not shown on the figure but a clear trend appears: the number of contributions quickly rose in the period 2000-2016 and has then decreased in intensity for the past 2-3 years. A possible interpretation is that WoT/IoT ontologies have reached maturity and a technology transfer occurred from research to the industry.

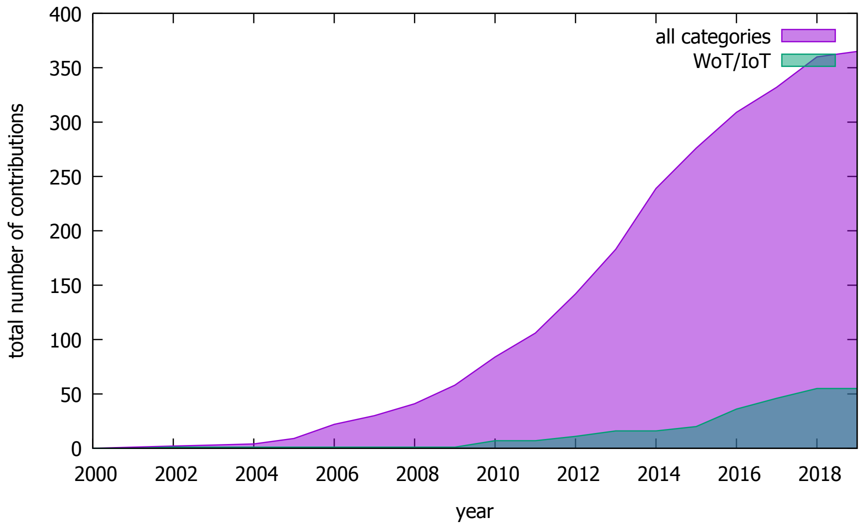


Figure 2: Evolution over time of the number of vocabularies registered in the LOV4IoT catalogue

This interpretation is otherwise supported by the fact that several of these ontologies have been standardized in the past years or are about to be standardized. The most important one is the SSN ontology, first incubated by the W3C [28] and now an official W3C recommendation [51]. SOSA, which defines the class `sosa:Sensor`, is a subset of SSN intended to be a leightweight ontology that can easily integrate with most IoT applications. SOSA also defines the classes `sosa:Observation`, `sosa:ObservableProperty` and `sosa:FeatureOfInterest`. In the

SOSA terminology, an 'observation' is a measurement at a given point in time of some 'property' of some 'feature of interest'. 'Feature of interest' is a very generic concept covering a variety of physical world entities. The same pattern is defined for `sosa:Actuators`. We come back to these concepts later in this chapter (Sec. 2.3.3).

SSN extends SOSA in several places to provide more context about sensor measurements, e.g. to specify system deployments and their capabilities in normal and degraded conditions. However, SSN remains an upper ontology insofar as some classes are placeholders for future alignment with domain-specific vocabularies. For instance, there is no definition for the concept of temperature in SSN, nor is there a definition of a binary switch, the most common IoT devices. SSN only includes the more abstract classes `ssn:Property` and `ssn:System`.

Several works on semantic interoperability and standardization in the IoT introduce an alignment of SSN with other vocabularies, in particular with the Smart Appliance Reference (model) (SAREF) ontology and the Smart Energy Aware System (SEAS) ontology [77, 87]. SAREF defines both classes `saref:Temperature` and `saref:BinarySwitch`, as well as other terms of in the domain of home and building automation. An alignment between SAREF and SSN would state that `saref:Temperature` is a sub-class of `ssn:Property` and `saref:BinarySwitch` a sub-class of `ssn:System`. SAREF was mostly driven by the industry and became a standard by the European Telecommunication Standards Institute (ETSI) [111]. SAREF mixes domain-specific and domain-independent concepts, it is therefore relevant to mention it in the present review. Indirectly, SAREF also aligns with the Ontology of Units of Measure (OM) by reusing some of its units. OM is not a standard but it is a comprehensive and well-maintained set of axioms.

The SEAS ontology provides both a generalization of SSN and several specializations (e.g. for the building automation, energy and environmental domains) [78]. The ontology was designed by active contributors of SSN and an alignment between the two ontologies is provided. A direct alignment between SEAS and SAREF was also proposed in the literature and an official endorsement of SEAS by ETSI is planned[2].

It is also worth mentioning a recent initiative to extend schema.org to the IoT[3], although only few classes and properties have been published so far. There are many more standardization bodies that are currently working on reference architectures, models and protocols for the IoT and WoT. Most of them are not included in the LOV4IoT catalogue because they rely on other technologies than RDF. Their development process is however similar in many respects and some of

2 https://portal.etsi.org/STF/stfs/STFHomePages/STF556
3 https://iot.schema.org/

the specifications they produce can be considered as shared conceptualizations, i.e. as ontologies. We review them next.

2.2.3 *Other Information Models*

At the application level, most communication standards include ontological definitions, although in an implicit fashion. Most standards indeed define an information model such that model elements directly map onto messages accepted or not by a protocol. WoT and IoT communication standards are therefore also relevant in the present study, although their model for 'things' and the medium to exchange information on these 'things' are interdependent. This is the case e.g. for the collection of (BLE) Generic Attributes (GATT), to expose information like automation I/O control or health measurements[4]. It is also the case for the widely used protocol for BA Control Network (BACnet)[5] and another widely used framework to communicate with field devices in industrial plants, the OPC Unified Architecture (OPC-UA)[6]. Both standards define their own object model with pre-defined object types.

Another category of standards, more specific to WoT, rely on the same Web protocol but they also restrict the set of messages that can be exchanged, in order to provide an integrated runtime environment that comes with features like authentication and device management. The main standardization bodies in this category are the oneM2M foundation[7], the Open Connectivity Foundation (OCF)[8] and the Open Mobile Alliance (OMA) in its set of standards on Light-Weight Machine-to-Machine (LWM2M) communication[9]. All 3 standards have a distinct meta-model, against which they define schemas for various sensor types (like 'temperature', 'accelerometer' or 'illuminance') and control interfaces (like 'light control', 'on/off switch' or 'up/down control').

Finally, although they are not specified alongside a protocol, some standards are used *de facto* in conjunction with specific architectures. For instance, the Electronic Device Description Language (EDDL) is used in the configuration of Profibus and Profinet devices in industry automation[10]. The standard product and service taxonomy eCl@ss[11] is also used to facilitate the deployment of industrial controllers in the

4 https://www.bluetooth.com/specifications/gatt
5 http://bacnetinternational.net/
6 https://opcfoundation.org/about/opc-technologies/opc-ua/
7 http://onem2m.org/
8 https://openconnectivity.org/
9 https://www.omaspecworks.org/
10 https://www.profibus.com/download/device-integration/
11 https://www.eclass.eu/

STANDARD	OWL MAPPING
BLE GATT	-
OPC-UA	-
BACnet	BACowl [8]
oneM2M	oneM2M Base Ontology [123], [26]
OCF	[26]
IPSO/LWM2M	[26]
EDDL	[57]
eCl@ss	eCl@ssOWL [56]
IFC	ifcOWL [90]
Project Haystack	HTO [21], Brick [6]

Table 3: Communication standards related to WoT with mapping of their information model to OWL

field. The Industry Foundation Classes (IFC)[12] and Project Haystack[13] found similar usages in Building Automation (BA) systems.

Table 3 provides a summary. Despite the fact that most standards are used in specific application domains, a certain overlap exists in the ontological models they define. In the first place, they all adopt an object model with read/write access on properties and object "methods" to execute certain tasks. For most of them, a mapping to OWL exists.

In the case of oneM2M and ifcOWL, this mapping is included in the specification of the model [90, 123]. The other available ontologies are experimental: eCl@ssOWL [56], EDDL mapping [57] and the Haystack Tagging Ontology (HTO) [21] (as well as on-going efforts around OPC-UA), to the exception of Brick, to which the Project Haystack community seems to be converging [6]. A recent work on comparing oneM2M, OCF and LWM2M in terms of semantic coverage also provides an indirect mapping to SSN and SAREF. Another relevant experiment was done with Konnex, another BA standard (also spelled KNX), although it consisted in embedding OWL declarations in a KNX frame, as opposed to mapping the frame structure of KNX to OWL [101].

All these OWL models theoretically increase interoperability across systems. Nonetheless, only few formal alignments have been proposed between them and other Web ontologies. First, an alignment between Brick and ifcOWL is indirectly available via the W3C Building Topology Ontology (BOT), currently being developed by the W3C community [98]. Second, the mapping from EDDL to OWL targets

12 http://www.buildingsmart-tech.org/specifications/ifc-overview
13 http://project-haystack.org/

the ontology for Quantity Kinds, Units and Data Types (QUDT), referenced by LOV4IoT [57]. The last alignment provided in the literature is between the oneM2M base ontology and SAREF [111]. Despite the absence of alignments with other standards, it is known that some of them overlap, both in terms of upper ontologies and domain-specific ontologies [26].

A synthesis between all the models listed in Table 3 is necessary to be able to introduce a generic definition for WoT 'things' that would take into account the agents that expose these 'things' using diverse protocols. To overcome the lack of formal alignments between these models, we must provide a comparison to LOV4IoT first, in particular to SSN, SOSA, SAREF and SEAS. We address this issue next.

2.3 WHAT IS A 'THING'?

We recall that the objective of the present review is to design an ontology for WoT in order to formally define the concept of 'thing'. We are therefore primarily concerned with ontological alignments between existing models, in order to select the most abstract concept that encompasses most if not all physical world entities that are either the subject or the object of sensing and actuation and whose representation is being exchanged via IoT and industrial protocols (like BLE, OPC-UA, BACnet and others).

So far, we have reviewed 58 Web ontologies and 10 other information models that potentially align. One of the outcomes of our review so far has been that there are too few alignments available for the latter, despite various attempts to provide a unified OWL-based view on these information models. There exists automatic or semi-automatic techniques to ontology alignment, a problem extensively studied in the literature also known as *ontology matching* [31]. Some of the well-known ontology matching techniques were used here to provide alignments between the non-RDF information models.

Web ontologies are designed to be modular such that a single ontology represents a consistent set of rules that are meant to be used together in a knowledge base. For this reason, alignments shall not be considered on a term basis but rather on a vocabulary basis. From Def. 1, it is possible to derive a graph of alignments, which can then be used to identify "reference" ontologies, to which most other ontologies align.

Definition 2. *The graph* $G = \langle V, E \rangle$ *is a graph of alignments for some ontologies* $\mathcal{K}_1, \ldots \mathcal{K}_n$ *if* $|V| = n$ *and* $\langle x_i, x_j \rangle \in V$ *if and only if* $\mathcal{K}_i$ *aligns with* $\mathcal{K}_j$ *as per Def. 1.*

In the following, two graphs of alignments are analized: one that can be constructed for non-RDF standards (Sec. 2.3.1) and another one for the ontology cloud of LOV4IoT (Sec. 2.3.2). The comparison results in a "bridging" ontology for the TD model, with edges pointing at both graphs (Sec. 2.3.3).

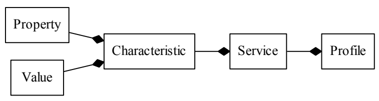
2.3.1 *Alignments Between Non-RDF Standards*

Because all standards listed in Table 3 use different formalisms, their main model elements had first to be extracted and represented in a simplified formalism (Fig. 3). The formalism that was chosen is a subset of UML that only features inheritance (▷) and composition (◆). The classes showed on Fig. 3 summarize only the meta-model of each standard. All model elements that are domain-specific were discarded, as was done in the review of LOV4IoT, to keep only the most abstract ontological level (upper ontologies). All ontological models included in these standards are much more extensive, though. As an example, BACnet specifies 54 object types that inherit the class Object and 38 service descriptions that inherit Service (Fig. 3c) [89]. Only the most common services are showed here: ReadProperty and WriteProperty.

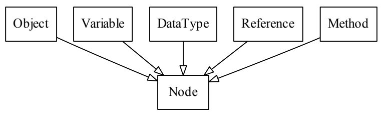
Given these sets of upper level classes, we can then find semantic equivalences between them. Among all ontology matching approaches, the simplest is to compare plain words and compute a syntactic similarity between them (e.g. an edit distance) [31]. For instance, we can see a class Object on four UML diagrams and Property on five of them (which is a mere consequence of the ubiquity of objects in computing). However, classes with the same name may have different meanings. It is the case for two of them: Resource and Service. A LWM2M resource is a Web resource while an IFC resource is most likely a "human resource" or some construction equipment. Similarly, a BACnet service is provided by a machine while in eCl@ss, it is to be understood as a service provided by a company.

To find relations between classes, all classes were mapped to sets of synonyms in WordNet, a lexical database in which plain words map to one or more such sets corresponding each to a given "sense" [33]. WordNet also provides semantic relations between senses that have correspondance with the present formalism: *hypernymy* (inheritance) and *meronymy* (composition). The mapping from UML to WordNet was defined such that the relations in either domain do not contradict (a form of structural ontology matching). Figure 4 shows the semantic equivalences that were found across standards, i.e. classes that map to the same sense in WordNet.

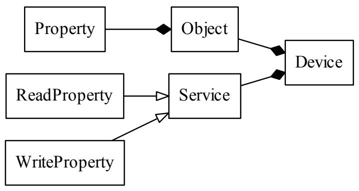
More advanced ontology matching techniques exist and the graph of alignments presented on Fig. 4 should not be interpreted as exhaustive but rather as "minimal". For instance, there is a certain overlap between the LWM2M class Execute and the class Method, defined among others by EDDL, and yet there is no edge between the two vocabularies in the graph. In fact, the graph only includes symmetric



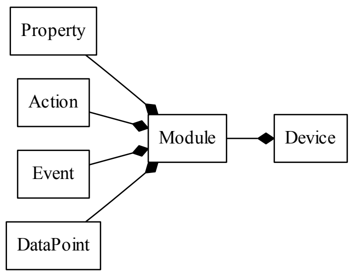
(a) BLE GATT



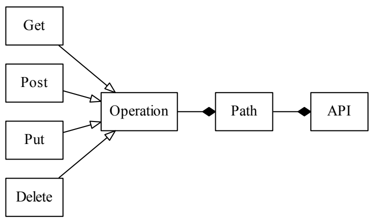
(b) OPC-UA



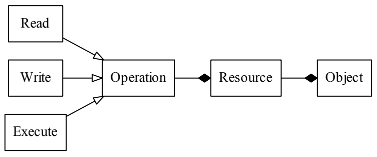
(c) BACnet



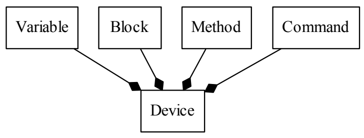
(d) oneM2M



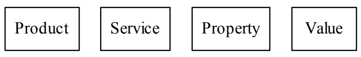
(e) OCF



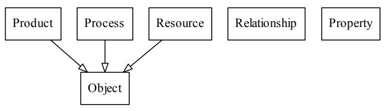
(f) LWM2M



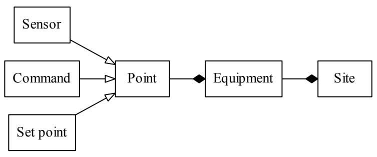
(g) EDDL



(h) eCl@ss



(i) IFC



(j) Project Haystack

Figure 3: Main concepts from the information model of the WoT-related standards of Table 3 (UML notation)

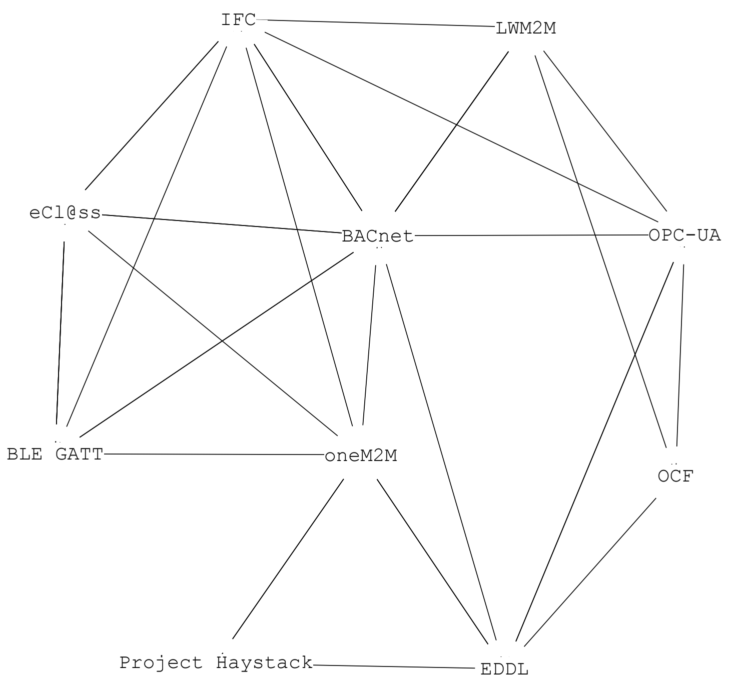


Figure 4: Graph of alignments constructed from lexical equivalences between the concepts represented on Fig. 3, aggregated per standard

equivalence relations, hence an undirected graph, although a graph of alignments may also have directed edges like between Execute and Method. Despite this limitation, we can still observe a significant ontological overlap between standards, which can be quantified by means of graph metrics: all standards align to at least two other standards and to 6.8 in average[14] (minimum and average vertex degree); alignments in the graph represent 76% of all alignments if the graph were complete (graph density) and the longest path between two standards is of size 3 (graph diameter).

Given a high density and a high vertex degrees in the graph of alignments one obtains, it is possible to identify the most representative standard(s) to use as a reference for formal alignments. If we take the maximum vertex degree as a measure of "representativity" or *centrality*, BACnet seems to be the best candidate (degree of 12). However, a more accurate measure would be the *farness* of each vertex based on the idea that the closer a vertex is to other vertices, the more central it is. Farness, the reciprocal to closeness, is computed as the average size of all paths from a given vertex to all other vertices in the graph. The standard with the lowest farness is oneM2M (farness of 11), followed by BACnet and IFC. Statistics for all vertices are reported in Table 4.

14 including multiple edges connecting the same vertices.

STANDARD	DEGREE	FARNESS
BLE GATT	7	1.6
OPC-UA	6	1.4
BACnet	**12**	1.2
oneM2M	8	**1.1**
OCF	3	1.7
LWM2M	6	1.5
EDDL	7	1.3
eCl@ss	7	1.5
IFC	8	1.2
Project Haystack	3	1.7

Table 4: Statistics on the graph of Fig. 4 with respect to vertices (WoT-related standards)

The oneM2M ontological model for WoT devices has the three classes Property, Action and Event (Fig. 3d). Interestingly, among the standards we have reviewed, it is the closest to the original W3C submission for a generic "WoT Thing Model"[15]. This submission, which initially used the term 'subscription' instead of 'event', developed into a versatile model for WoT that uses exactly these three classes, while oneM2M's DataPoint is subsumed by the W3C model's Property class [61]. Arguably, this systematic approach to semantic alignment based on graph metrics formalizes the more intuitive approach of the W3C WoT community to develop its model, also known as the TD model. The present analysis guarantee by no means that this ternary pattern with Property, Action and Event is the only possible model for WoT but it guarantees at least that it is a correct one, in the sense that all WoT and IoT standards can semantically align to it. An alternative approach that would also be correct is to adopt the more classical object models of BACnet or IFC.

By showing the centrality of oneM2M, one indirectly shows the centrality of the W3C TD model as well, assuming that lexical alignments in WordNet can translate in the pure semantic domain. However, if this (simple) TD model is a good basis for a TD ontology, it does not make for an ontology alone. It defines indeed the concept of 'thing' in a very broad sense, in no relation to other concepts [64]:

THING an abstraction of a physical or virtual entity whose metadata and interfaces are described by a WoT Thing Description.

In other words, a 'thing' is anything that has a TD document, i.e. a formal description of itself accessible on the Web. This definition is

15 https://www.w3.org/Submission/wot-model/

close to that of `owl:Thing` as it does not restrict to sensing or actuation. To refine this definition, one should provide further alignments between the TD model and other Web ontologies like SSN, SOSA, SAREF and SEAS, which have already been identified as emerging references for WoT (Sec. 2.3.2). In the following, more insights are given on LOV4IoT by providing the same analysis as for non-RDF standards on the whole LOV4IoT ontology cloud.

2.3.2 *Alignments in the Web of Things Ontology Cloud*

Unlike the simplified UML models that was presented in the last section, LOV4IoT ontologies themselves provide alignments to other ontologies. These alignments go beyond the strict equivalence of last section, as formalized in Def. 1. For instance, the original LOV platform distinguishes between five types of alignments other than equivalence: import, extension, specialization, generalization and disjunction [117]. In the scope of this thesis, it suffices to know that alignments can be automatically extracted from OWL documents on a purely syntactic level. A graph of alignments for LOV4IoT can be constructed on this basis, similar to the one that was presented in last section. An earlier graph of alignments for LOV4IoT was already published in 2016 but the present version takes into account changes to the LOV4IoT catalogue since then [24].

Web ontology documents can import other documents that are not necessarily referenced by LOV4IoT. From the 58 contributions in the categories of 'IoT' and 'WoT', 78 documents were crawled from 42 contributions. Missing documents were either not made public or could not be parsed properly. Some of the crawled documents also included alignments with other ontologies without explicitely importing them. These were also included in the graph, for a total of 85 documents (each in its own namespace) with 285 alignments. Figure 5 shows the final result[16].

If we first consider this graph as indirected, the same graph metrics as for non-RDF standards can be reused (Fig. 4). In comparison, it is much more sparse: its density is of 10%, its average degree is 3.7 and it has a diameter of 7 (if considered as undirected). Then, when we include edge directions, what this graph highlights is the dominance of SSN as a reference ontology in most contributions. Indeed, it has a much higher in-degree than any other ontology (27 directed edges point to it, twice as much as 95% of the vertices) and it also has by far the lowest farness value. Since the graph is directed, one can also consider PageRank as a measure of popularity among ontologies [16]. Again, SSN stands out as the vertex with the highest PageRank.

16 for clarity, related ontology modules have been grouped and self links have been removed.

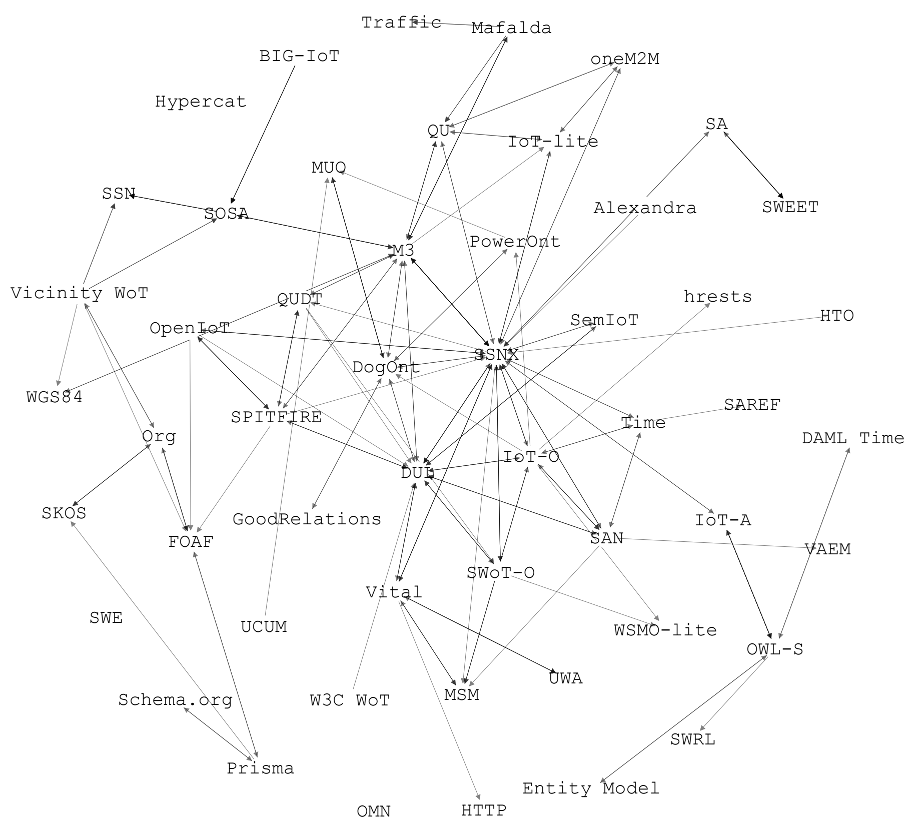


Figure 5: Graph of alignments extracted from Web ontology alignments declared within LOV4IoT

ONTOLOGY	IN-DEGREE	FARNESS	PAGERANK
SSNX	**27**	**1.875**	**0.039067**
DUL	20	2.167	0.023299
M3	13	2.23	0.022743
SOSA	8	2.938	0.006796
DogOnt	8	2.417	0.024721
QU	7	2.604	0.013021
MSM	6	2.667	0.008613
FOAF	6	3.021	0.015887
SSN	6	3.75	0.005725
SPITFIRE	5	2.375	0.009752
IoT-O	5	2.5	0.013867
SAN	5	2.625	0.00999
Vital	5	2.708	0.015621
OWL-S	5	3.542	0.017288
SWEET	5	3.792	0.005171

Table 5: Statistics on the graph of Fig. 5 for the 15 vertices with the highest in-degree; tampering parameter of PageRank $p = 0.85$

Statistics for LOV4IoT are reported in Table 5 for the 15 vertices with the highest in-degree. LOV4IoT does not guarantee that the provided semantic alignments are consistent but this pattern we observe on all three metrics suggests that SSN is good enough to be used as a basis for alignment. The table also shows the importance of Dolce+DnS Ultralite (DUL), an upper ontology with a much broader scope than SSN. In its original version (SSNX), SSN even aligns to DUL. It is also worth noting that most contributions align to SSNX and not to the final W3C standard. The former was incubated by the W3C and did not significantly change throughout standardization, though.

The other ontologies listed in Table 5 cover various domains: the Machine-to-Machine Measurement (M3) ontology and the IoT Ontology (IoT-O) provide a generic IoT model [50, 107], DogOnt was the first extensive research project for home automation and domotics [11], the ontology for Quantity and Units (QU), a subset of QUDT, and the Semantic Web Earth and Environment Terminology (SWEET) ontology cover physical quantities and their units of measure [96, 97], the Minimal Service Model (MSM) and the OWL for Web Services (OWL-S) are used to annotate Web services [85, 91], the Friend Of A Friend (FOAF) ontology for social networks is one of the first ontologies on the Web [15], SPITFIRE and Vital are applications-specific models [63, 93] (we will come back to SPITFIRE in Ch. 4) and finally, the Semantic Actuator Network (SAN) is a translation of SSNX to actuators, although it is now subsumed by the more recent SSN [107].

From this list, we can already see overlaps in terms of domain coverage. In particular, LOV4IoT includes redundancies for physical quantities and units and Web service descriptions. In addition to QUDT and SWEET, some ontologies also reference OM and an OWL variant of the Unified Code for Units of Measure (UCUM). Moreover, DogOnt and SAREF redefine some quantity kinds to suit their own need, which leads to many implicit equivalence relations between classes. Any of QUDT, SWEET and OM could become a reference as they all are extensive and well-maintained ontologies.

Regarding Web service descriptions, there are redundant conceptualizations between OWL-S and the Web Service Modeling Ontology (WSMO), referenced by IoT-O [32]. Attemps to capture the intersection of both formalisms like MSM seem however to fail to become a reference. In addition, none of these ontology fully captures the semantics of RESTful Web services, which are an important part of WoT [73]. RESTfulness is indeed the foundation of frameworks like oneM2M, OCF and LWM2M.

We can conclude this analysis of LOV4IoT in two points: first, an alignment with SSN will capture most of the concepts defined in the WoT and IoT research community and, second, a choice is to be made on what ontologies to use to model quantity kinds (along with their units of measure) and Web services exposed by WoT agents. For these two aspects, SSN only defines (abstract) placeholder classes: Property and Procedure. Aligning the W3C TD model with SSN would bridge non-RDF communication standards and LOV4IoT: an alignment is presented in the following.

2.3.3 *The Thing Description Model*

The W3C TD Model is a simple model close to that of oneM2M [61]. Recall the definition for the class Thing (Sec. 2.3.1): a 'thing' is anything described by a TD document. A TD document is in fact a collection of *affordances*, that is, of hyperlinks a WoT agent can follow to access sensor and actuator data. These affordances, materialized by the class InteractionAffordance, expose the interaction endpoints of the servient managing this data (another WoT agent). They are of three kinds: 'read/write property' affordances (Property), 'invoke action' affordances (Action) and 'subscribe to event' affordances (Event). They are analogous to object properties, methods and event listeners in object-oriented programming. A definition of each concept is provided in the following, derived from the W3C definitions:

INTERACTION AFFORDANCE Information entity providing the necessary knowledge to interact with the servient exposing it. Every interaction affordance is associated to a ‘form’ (similar to a Web form).

PROPERTY (READ/WRITE AFFORDANCE) Interaction affordance to read and/or write some property of a ‘thing’ characterizing its internal state. The state of a ‘thing’ may not be fully characterized by its properties (hidden state).

ACTION (INVOKE AFFORDANCE) Interaction affordance to invoke a procedure (or action) to act directly or indirectly on the internal state of a ‘thing’. An action may also be a pure function (no state change). The duration of an invoked action may be much longer than the duration of the invokation itself, in which case the affordance should include the necessary knowledge to monitor the invoked action.

EVENT (SUBSCRIBE AFFORDANCE) Interaction affordance to listen (or subscribe) to an event thay may occur once or repeatedly and to cancel that subscription. An event may represent a change in the internal state of a ‘thing’.

These definitions adopt a “systemic” view on ‘things’ where the concept of ‘thing’ conflates with that of ‘servient’. It is indeed quite natural to consider that physical world objects have a digital representation only if some system can observe and act on their properties. In fact, oneM2M adopts a similar definition for ‘thing’ by defining the class Device as a sub-class of Thing in its base ontology. However, it is sometimes beneficial to hide a system from external servients and directly expose a representation of some physical world entity, in which case the entity is a ‘thing’ itself. Among others, this extension of the definition of ‘thing’ is consistent with Kevin Ashton’s coinage of the term ‘IoT’: when a manufactured object is added an RFID tag, there is often no need to semantically model the tag itself but rather the object[17].

A formal alignment between TD classes and SSN can now be introduced. As every servient can be considered as a system, aligning Thing with System is straightforward. As per last remark, Thing should also align to Platform, a class designating anything hosting a system, like a manufactured product hosting an RFID tag. Finally, because of the structural similarity, a Thing can align with the class FeatureOfInterest, which designates any physical world object with observable and actuatable properties. Some Systems are themselves instances of FeatureOfInterest, in particular when they have the ability to expose their own internal state as properties. The OWL formalization of this alignment is presented on Fig. 6, using the Visual notation for OWL (VOWL) [81].

With this formalization, we part from the ambiguity between `owl:-Thing` and `td:Thing`. If we consider the union of System, Platform and

17 we reported this observation to the W3C WoT working group, which adopted the extension being exposed here of the definition of ‘thing’.

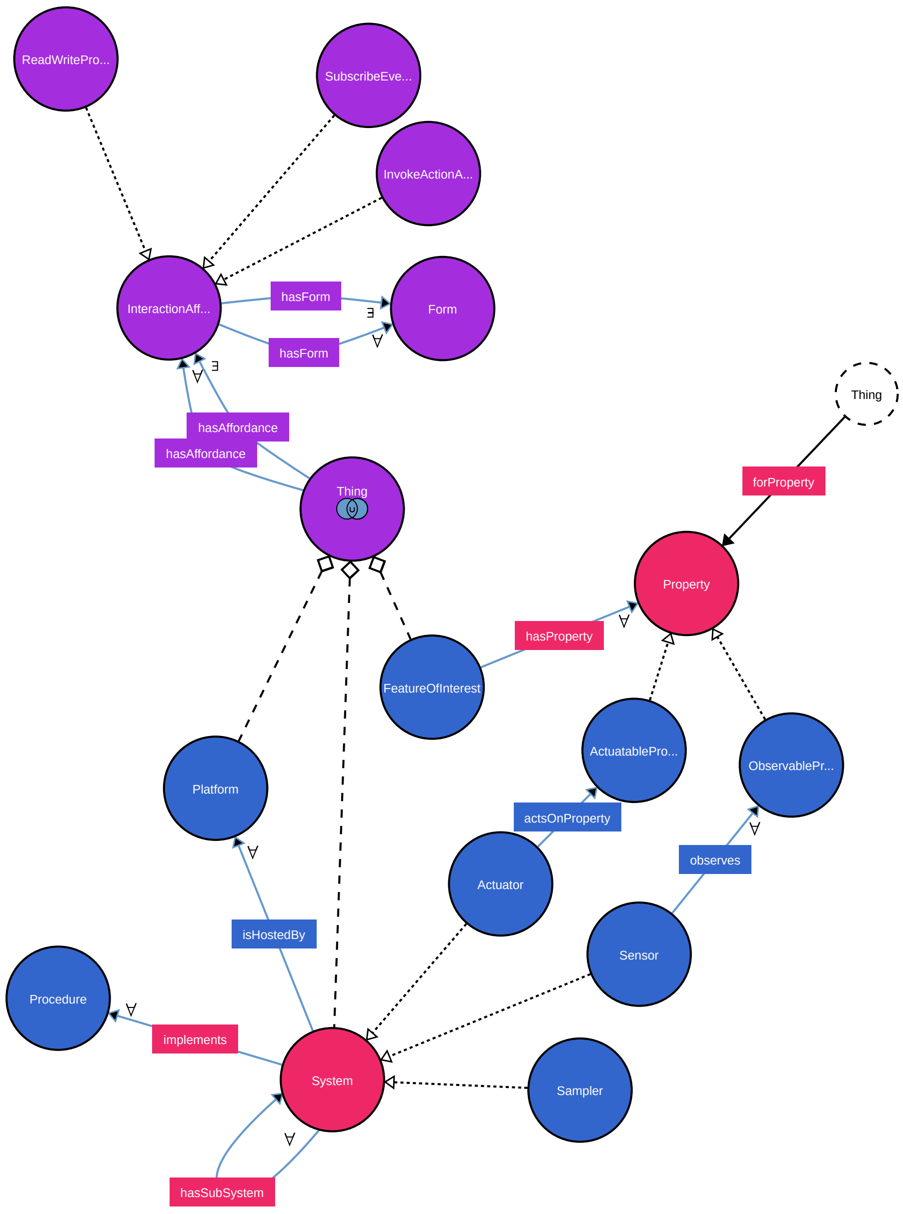


Figure 6: Main concepts of the TD ontology with alignment to SOSA & SSN (VOWL notation); label colors encode namespaces (purple → TD, blue → SOSA, red → SSN)

FeatureOfInterest as exclusive, it then follows that a Property or a Procedure cannot be a `td:Thing` while it is still an `owl:Thing`. As an attempt to provide a less ambiguous synonym for the former, the term *tangible object* may as well be a good fit. By opposition, it is anything that is not Intangible, as defined by schema.org. The class PhysicalObject, defined in the Dublin Core ontology[18], shares some aspects with the present definition. Dublin Core is one of the most widely used Web ontologies.

Given this alignment between TD and SSN, examples of 'things'can be provided, either by looking at instances of one of the three SSN classes or by looking at their sub-classes in LOV4IoT. The W3C SSN specification provides numerous examples[19], according to which the following entities are potential instances of Thing[20]:

- sensors & actuators:
  - an atmospheric pressure sensor
  - a temperature and humidity probe
  - an energy consumption meter
  - a window closer
  - a laser range finder
  - a seismograph
  - a wind sensor
- other computational or electronic devices:
  - an iPhone
  - a Java SunSpot device
  - a printed circuit board
  - a LoRa communication device
- physics- and biology-related entities:
  - the atmosphere of Earth
  - a tree
  - Earth
  - the Antarctic ice sheet
  - air in the area of the Metropole Lyon
- manufactured items:
  - a window
- places:
  - the Coal Oil Point reserve
  - a room

Among manufactured items, one can also think of more complex machinary constructed as an assembly of simpler parts. The follow-

18 `http://dublincore.org/`
19 `https://www.w3.org/TR/vocab-ssn/#examples`
20 the classification introduce here is for the mere convenience of the reader, it does not aim at exhaustivity nor is it a strict conceptualization.

ing entities also present in the SSN examples shall *not* be instances of Thing:

- a temperature
- a speed
- the Metropole Lyon (as a human organization, not a place)

If we look at classes in LOV4IoT and other standards, we can observe that most examples have a correspondance. For instance, the SAREF class Device has the sub-classes TemperatureSensor and EnergyMeter. It is possible to construct a class for any kind of sensor and actuator by using e.g. the classes Pressure, Humidity, Speed or Distance defined both in OM and QUDT. Most physics-related entities have a corresponding class in SWEET. For places, schema.org provides the class Place and WGS84, a minimal ontology to specify geo-locations also provides the class SpatialThing. BOT, ifcOWL and Brick have an abstraction for rooms, windows and other notions related to buildings (Space, Element). Finally, eCl@ssOWL can be used to refer to manufactured items. In Ch. 1, we had a look at yet other examples of 'things'. It is now possible to provide a complete RDF representation of these 'things' on the basis of LOV4IoT definitions.

Example 3. *The water management plant model we saw in the beginning of the thesis (Fig. 1) includes the 'things' e104, b104, v102, s115 and s116, which can be described as follows (using eCl@ssOWL, OM and SOSA):*

```
temp:Temperature.
level:Height.
tank:WaterTank[hasProperty → temp, hasProperty → level].
e104:Heater[isHostedBy → tank, actsOnProperty → temp].
b104:TemperatureSensor[isHostedBy → tank, observes → temp].
v102:PneumaticValve[
  actsOnProperty → level,
  hasSubSystem → s115:FloatSwitch,
  hasSubSystem → s116:FloatSwitch
].
```

Example 4. *This thesis was introduced with an anecdote on Stuxnet. Here is an RDF representation of the devices that were targeted by the Stuxnet program (using eCl@ssOWL, WGS84 and SOSA), for* $i \in [1,6]$ *and* $j \in [1,64]$ *:*

```
plant:System[
  hasSubSystem → cascade_i:System[
    hasSubSystem → plc_{i,j}:FrequencyConverter[
      isHostedBy → centrifuge_{i,j}:Centrifuge
    ]
  ].
].
```

In the remainder of the thesis, the subset of the ontologies cited in this chapter that align with TD via SSN will be referred to as the "WoT ontology cloud". In this ontology cloud, OM was chosen over QUDT as reference for physical quantities and units, given that it directly aligns with SAREF. This cloud does not include any ontology designed for Web service description, given that interaction affordances and the associated Form class already play that role. The complete list is provided in Appendix B.

2.4 NOTE ON USING WEB IDENTIFIERS FOR 'THINGS'

One immediate issue to address after giving a definition for 'things' is to find an identification scheme for all 'things' on the Web. Indeed, since the definition introduced here covers not only computational systems but also physical platforms and features of interest, it is not enough to rely on network addressing mechanisms, e.g. at the IP level, to identify them. To give 'things' a Web presence, they must be identified with an IRI. The relevant question here is therefore to find URI schemes are appropriate for that purpose.

As a matter of fact, the question of identification on the Web has regularly been the subject of discussions since the creation of the World Wide Web[21]. At the time the idea of a Semantic Web emerged, refinement in the definition of an IRI was required: any physical world entity could then be given an IRI, at which an RDF description would be exposed. Solutions to the question of identity on the Web even anticipated the idea of extending the Web to sensors and actuators [95].

It quickly became clear that a physical world entity and a Web resource providing a representation for it had to have distinct identifiers. As an example, the Wikipedia page about Munich is not Munich itself. The former has e.g. a creation date but no geo-location (as a document) and, conversely, the latter has no creation date but it has a geo-location (as a place). In WoT, a single 'thing' may be described

21 the outcomes of a WWW workshop named *Identity, Reference and the Web* in 2006 served as a starting point for this section. See http://www.ibiblio.org/hhalpin/irw2006/.

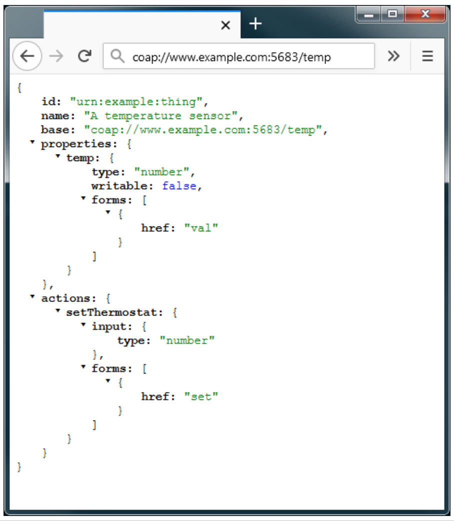
by two distinct TD documents, served by different servients or even by "proxy" servers that are not themselves WoT agents. If all TD documents must have their own IRI to be dereferenceable, the IRI of the 'thing' must be location-independent. One pattern that emerged around (Semantic) Web resources involves three components: an 'identifier', the 'resource' and the 'entity' [37]. This pattern, called IRE, was the basis for an initial design of a WoT ontology that differentiates between the classes `Thing` and `ThingDescription` [24]. As this classification suggests, the 'thing' is the entity while the corresponding TD document is the resource. The TD document, which is exposed by a servient, must have a dereferenceable URL, which is then the IRE identifier. In fact, the latter rather plays the role of a "locator" for Web browsers to download and process the underlying resource. for an illustration of IRE for a 'thing'. On this example, the micro-controller chip exposes a TD document describing itself at `coap://example.org/temp`. A Web browser can browse to this location and process the document, as shown on the screenshot. However, it is transparent to the browser what IRI was used to identify the physical chip itself (in this case, `urn:example:thing`). If the IRE pattern provides a clean separation between the Web resource and the physical world entity, it does not directly address the question of identifying the entity itself.

Official guidelines developed within the W3C exist for the naming of physical world entities that still allow an agent to get a representation of these entities [105]. The two suggested approaches rely on HTTP IRIs, either by using an IRI fragment to identify the entity (such that an HTTP request will still succeed) or by using HTTP redirection to redirect from the entity IRI to another IRI identifying a Web resource, like a TD document for some 'thing'. In the context of WoT, these approaches come with numerous issues with respect to scalability, security and lifecycle management, given the high number of 'things' a WoT system may deal with. Among others, validating the content of a TD document would require to communicate with potentially untrusted third-party servers (that either serve base resources or perform redirection).

An alternative approach would be to use reserved URI schemes for non-dereferenceable identifiers, like `urn:` for a Uniform Resource Namespace (URN). A list of URI schemes and URN namespaces compatible with WoT are provided in Tables 6 & 7. Only a subset of possible schemes[22] and namespaces[23] were considered here, namely those officially registered by the International Assigned Numbers Authority (IANA), that manages Internet domain names. This list is not exhaustive; to give one more example, one could think of `mailto:` to

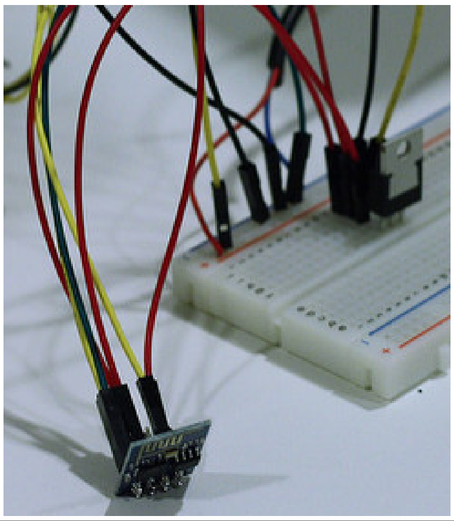
22 https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml
23 https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml

`coap://example.org/temp`	Identifier



```
{
   id: "urn:example:thing",
   name: "A temperature sensor",
   base: "coap://www.example.com:5683/temp",
 ▾ properties: {
   ▾ temp: {
        type: "number",
        writable: false,
      ▾ forms: [
        ▾ {
              href: "val"
          }
        ]
     }
   },
 ▾ actions: {
   ▾ setThermostat: {
      ▾ input: {
           type: "number"
        },
      ▾ forms: [
        ▾ {
              href: "set"
          }
        ]
     }
   }
}
```

Resource



Entity

Figure 7: Illustration of the Identifier-Resource-Entity pattern

URI SCHEME	EXAMPLE
URN	`urn:example:thing`
Named info.	`ni:///sha-256;f40xZX_x_FO5Lc...`
Information	`info:example/thing`
Distributed id.	`did:example:123456789abcdefghi`
IoT discovery	`iotdisco:NAME=thing;MAN=example.org`
Tag	`tag:example_thing`
Ext. resource id.	`xri://example.com/(urn:example:thing)`

Table 6: List of URI schemes suitable to identify 'things' (not exhaustive)

URN NAMESPACE	EXAMPLE
Electronic product code	`urn:epc:id:sgtin:900100.0003...`
Universally unique id.	`urn:uuid:f81d4fae-7dec-11d0...`
Device id.	`urn:dev:mac:0024befffe804ff`

Table 7: List of URN namespaces suitable to identify 'things' (not exhaustive)

identify humans (who are potential Platforms and therefore potential Things).

At first sight, non-dereferenceable IRIs may have the shortcoming not to be self-contained: the IRI alone is not enough to retrieve information about the resource it identifies, which goes against the core principles of RDF. To that remark, one can first oppose that certain URI schemes or URN namespaces specify well-known online catalogues such that IRIs may be indirectly dereferenced. It is the case e.g. for the International Standard Book Number (ISBN) and Global Trade Identification Number (GTIN) standards. Second, in the particular context of WoT, it is very likely that servients obtain IRIs for physical world entities only via TD documents, which must be retrieved before any interaction can take place. There should therefore be a dereferenceable resource IRI associated to every 'thing' entity on WoT, as per the IRE pattern. In addition, the W3C WoT specification states that all TD documents *must* provide links to Web resources for all interaction affordances.

2.5 SUMMARY

A first effort in defining semantics for WoT was to identify what *vocabulary* to use to describe WoT servients and their environment in a knowledge graph. To this end, 58 Web ontologies referenced by the LOV4IoT catalogue were reviewed. Most contributions to LOV4IoT come from a community of researcher but some notable ontologies are maintained (or will be) by standardization bodies: SOSA, SSN, SAREF and SEAS. The chapter also includes a review of 10 standards not based on RDF, like BACnet and oneM2M. Most of these standards have been aligned to RDF, although not in a unified manner, which prevents a direct comparison with LOV4IoT.

Two of the models that were reviewed are my own contributions: HTO [21] and the RDF integration of oneM2M with OCF and LWM2M [26]. However, the objective here is not to compare models with each other but rather to find possible alignments, in order then to provide a definition for the concept of 'thing' that would subsume all these models. LOV4IoT and the 10 non-RDF models that were reviewed have then been compared in terms of *graphs of alignments*, in which edges represent alignments between two models. A graph was first built for the non-RDF models by manually finding lexical relations between their main concepts. The node with the highest *centrality* value in this graph is oneM2M, which turns out to be similar to the TD model developed by the W3C (although other object-oriented models like that of BACnet would also have been suitable). The same operation was performed for LOV4IoT, in which SSN (another W3C standard) is unambiguously the most central node. This work significantly extends an earlier analysis, by myself, of LOV4IoT [24].

It follows from these conclusions that an alignment between the TD model and SSN would make for a generic model for 'things' which can theoretically align to all ontological models that exist in the literature. Consequently, an alignment in wich the class Thing is the union of the SSN concepts of System, Platform and FeatureOfInterest was presented. The latter concepts are different insofar as Systems are the "active" subject in a measurement or actuation process while FeatureOfInterests and Platform are "passive" objects, hence the further abstraction of all these entities being Things involved in some cyber-physical system. A number of examples of Things based on this alignment can be found in this chapter, like in Exs. 3 & 4.

Because not only computational devices are included in the definition but any tangible entity like manufactured items and places, one issue to address is the identification on the Web of these entities. Short guidelines were provided on what URI schemes to use to refer to such entities. These guidelines do not mention in which context to use these identification schemes and, in fact, there are numerous examples of 'things' for which no obvious choice exists. In practice, it is likely that not all entities relevant in a WoT system have an IRI that can be shared among WoT servients. We must therefore also take "anonymous" entities into account, whose existence can be stated but that cannot be fully identified. Next section formally introduces semantics for WoT, with particular care given to these anonymous entities.

3 REASONING WITH THING DESCRIPTIONS

3.1 INTRODUCTION

In the general introduction to the thesis, what was considered is the question of *identifying* physical world objects (Sec. 1.2.2). Most approaches to this problem in the IoT consist in providing methods for digitally “tagging” physical world objects with electronic chips. We have already seen examples of relevant technologies in that respect: RFID via Near-Field Communication (NFC), Zigbee, BLE. These technologies provide short-range communication protocols to efficiently expose few bytes of data at an unprecedented scale. As a result, electronic chips have become the successor to bar codes, to be used to not only identify consumer goods but all sorts of ‘things’. Alternative technologies have also been used for that purpose: in BA commissioning, an interesting practice to identify lighting devices has consisted in encoding byte sequences as light pulses at a frequency that is invisible to humans but that can be captured by camera-equipped terminals.

What these approaches have in common is that they specify a unidirectional transformation from the physical world to the digital space. If the target of this transformation is a set of IRIs as described in Sec. 2.4, then physical world objects immediately get a Web presence. The term Physical Web is sometimes used to refer to the result of this transformation[1]. However, a collection of IRIs is arguably not enough to make a “web” of Things. Tagging cannot provide links between resources, e.g. between a temperature sensor and the water tank on which it is mounted (like `b104` and `tank` in Ex. 3) or between a radiator and the room in which it is located. These links are *contextual*, they refer to how system components were deployed rather than to their own properties.

In Ch. 2, an object-oriented formulation of logical assertions was introduced, grounded in RDF, by means of which one can describe ‘things’ and their properties. In the generic TD model, ‘things’ are first described in terms of affordances to observe or act upon their properties. As WoT systems are defined in the thesis (Sec. 1.2.3), servients follow these affordances exposed by other servients to perform given automation tasks. However, in the general case, affordances alone do not suffice: not all servients on the Web must read the temperature exposed by `e104`, only those also mounted on `tank`. To find possible interactions in a WoT system, contextual information is also necessary.

1 https://google.github.io/physical-web/

This chapter develops a semantic framework for WoT based on collections of TD documents that embed this kind of information. The core principle of this framework is the following: the relations that exist between 'things' in a collection of TD documents, i.e. a knowledge graph, determine the interactions that can take place between servients exposing affordances for these 'things'. However, relations between 'things' are not necessarily explicit. They sometimes follow from other assertions in TD documents. For instance, if a radiator heats some room, it also heats adjacent rooms if their interface is permeable to air. Similarly, the water level of tank is acted upon by any valve that belongs to the same network of tanks and pipes (like v102). The two examples above are generalities often referred to as *domain knowledge*. Web ontologies are likely to contain domain knowledge, encoded in the form of rules. It is the case for several ontologies of the WoT ontology cloud of last chapter, like ifcOWL and BOT in the BA domain.

Rules found in Web ontologies can then be processed in a reasoning framework to extend the knowledge graph composed of TD documents. To extend a knowledge graph means to infer either new edges or new vertices. As we will see in a review of the state-of-th-art, most of the computational logic literature includes works that consider the simpler case of inferring edges only. However, in the context of WoT, it is more realistic to also allow for the inference of new vertices, like properties of a physical body or parts of an industrial system. The latter is referred to as *existential reasoning*. Indeed, it is only possible to infer "anonymous" vertices which simply account for the existence of some physical world entities, without identifying them with an IRI.

This chapter starts with a review of the state-or-the-art in existential reasoning with Web ontologies (Sec. 3.2). Some of the techniques found in the literature are then extended to discover potential interactions in a WoT system, from which general semantics for WoT can be derived (Sec. 3.3). This approach was tested on two different use cases, in the BA domain with a dataset provided by Intel labs (Sec. 3.4.1) and in the domain of industry automation, on the water treatment plant model introduced in an earlier chapter (Sec. 3.4.2).

3.2 RELATED WORK: EXISTENTIAL REASONING

Broadly speaking, reasoning is the process of drawing conclusions from certain knowledge [79]. Reasoning is not a particular problem-solving or decision-making task but rather a tool to complete such tasks. In WoT, reasoning can help agents build a consistent model of the physical world, either by validating sensor observations or by inferring high-level statements from the discrete observations they make of their environment. In the present case, reasoning can also make relations between 'things' explicit to drive servient interactions.

In computational logic, every reasoning task can be reduced to the problem of *entailment* of a assertion (or an axiom) α by a set of assertions and rules (a knowledge base) $\mathcal{K}$. The notation $\mathcal{K} \models \alpha$ expresses that $\mathcal{K}$ entails α. A WoT knowledge base would typically include axioms provided by some TD document for specific 'things', as well as generic knowledge about the physical world provided by Web ontologies. As mentioned before, the reasoning task of interest in WoT is that of existential reasoning. The two main existential reasoning techniques in use in the Semantic Web are being reviewed in the following: RDF graph canonicalization (Sec. 3.2.1) and existential reasoning with Web ontologies (Sec. 3.2.2). We then move on to the non-standard reasoning tasks of query answering and abduction (Sec. 3.2.3).

3.2.1 *RDF Graph Canonicalization*

The RDF data model includes the notion of *blank node*. According to the *RDF 1.1 Semantics* specification document [54]:

> Blank nodes are treated as simply indicating the existence of a thing, without using an IRI to identify any particular thing.

In other words, it is possible to provide axioms about unknown physical-world entities using blank nodes. The RDF simple semantics provides a means to compute equivalences between blank nodes and other entities, which is a form of existential reasoning. These equivalences are established by computing a canonical form G_C for an RDF graph G such that $G_C \models G$ and all blank nodes in G are replaced by "fresh" IRIs (not present in G) called Skolem constants. The procedure of merging equivalent nodes under the same IRI is called *leaning*.

An algorithm based on node coloring with graph leaning has recently been developed for canonicalization, [58, 59]. The following example briefly illustrates its principle.

Example 5. *Consider the following description of two radiators in room 31.638 of the Siemens Legoland campus, expressed in terms of BOT classes and properties:*

```
legoland:Site[
  hasSpace → 31.638:Space[
    containsElement → :Radiator,
    containsElement → :Radiator
  ]
].
```

Note that two individuals have a class (Radiator) but no identifier: they are blank nodes. Node coloring consists first in assigning an arbitrary label (a color) to every blank node in the graph and then blending these colors according to the neighborhood of each node in terms of classes and properties. Since both nodes describing a radiator have the same neighborhood, they will eventually be merged into a single node during canonicalization. Indeed, the existence of two radiators necessary implies the existence of one radiator; blank nodes do not carry any notion of cardinality.

One way to distinguish between the two radiators is to precise on which wall they are mounted (among other kinds of discriminative features).

Example 6. *Let us update Ex. 5 with South and East walls.*

```
legoland:Site[
  hasSpace → 31.638:Space[
    containsElement → :Wall[
      hasOrientation → south,
      hasSubElement → :Radiator
    ],
    containsElement → :Wall[
      hasOrientation → east,
      hasSubElement → :Radiator
    ]
  ]
].
```

Given the information that the two radiators are mounted on walls with different orientations (south, east), node coloring will output different colors, from which one can conclude that they are indeed distinct entities.

In practice, RDF graph canonicalization is a fragile reasoning framework. Adding a property to a blank node will have effects on all connected blank nodes in graph G. Moreover, node coloring does not include any kind of background knowledge. For instance, if we have 31.638[containsElement → :Wall[hasSubElement → :Radiator]], then, transitively, it is also true that 31.638[containsElement → :Radiator], according to a rule formalized in BOT. If such implicit statements from G are made explicit in G', then we have the undesired property that $G'_C \neq G_C$. Next, we consider existential reasoning in the presence of logical rules, as provided by Web ontologies.

3.2.2 *Reasoning with Web ontologies*

3.2.2.1 *Description Logics*

As already mentioned in Ch. 2, the standard language for Web ontologies is OWL. Its theoretical foundations are Description Logics (DLs), a family of logic formalisms developed in parallel to classical First-Order Logic (FOL). The following section introduces formal definitions for DL knowledge bases and develops the interplay of DLs with FOL and logic programming. Although the DL literature has developed its own terminology, slightly different from that of OWL[2], OWL terms will be retained throughout this chapter.

In all definitions, N^C, N^P and N^I respectively denote a set of class names, property names and individual names (all entity names are IRIs) and V denotes a set of first-order variables. A DL knowledge base designed for existential reasoning can now be formally defined.

Definition 3. *[69] Let* $t, t'_1 \dots t'_n \in V \cup N^I$, *let* $p_1 \dots p_n \in N^P$ *and* $C, C'_1 \dots C'_k \in N^C \cup \{\top, \bot\}$. *A DL expression is a formula* f *of the form*

$$t{:}C[p_1 \Rightarrow C'_1, \dots p_k \Rightarrow C'_k, p_{k+1} \rightarrow t'_{k+1}, \dots p_n \rightarrow t'_n].$$

A DL knowledge base is a set of rules $H :- B$ *where* H *(the rule head) is either of the form* $t{:}C$, $t[p \Rightarrow C']$ *or* $t[p \rightarrow t']$ *and* B *(the rule body) is a conjunction of DL expressions such that*

- B *is* tree-shaped *if seen as an undirected graph and*
- *if* H *is of the form* $t[p \rightarrow t']$, *then there is no path from* t' *to* t *in* B.

The top class $\top$ (respectively, bottom class $\bot$) is a special class defined as the super-class (respectively, sub-class) of every class in N^C. Formally, we have the rules $x{:}\top :- x{:}C$ and $x{:}C :- x{:}\bot$ for all $C \in N^C$ and for all knowledge base. Moreover, the body of a rule can be empty to express facts that always hold, i.e. assertions. For a given knowledge base, the set of rules of this form is called an ABox while other rules belong to what is called a CBox.

Example 7. *Here is a simple example of knowledge base* $\mathcal{K}_{ex}$, *stating that every space in a building is a body of water and every physical body, including air, has some temperature property:*

$$\begin{aligned} x{:}\mathit{Air} &:- x{:}\mathit{Space}. \\ x{:}\mathit{PhysicalBody} &:- x{:}\mathit{Air}. \\ x[\mathit{hasProperty} \Rightarrow \mathit{Temperature}] &:- x{:}\mathit{PhysicalBody}. \end{aligned}$$

2 in particular, classes and properties are respectively called concepts and roles in the DL literature.

The double-arrow ($\Rightarrow$) expresses an *existential restriction* on the class $\mathsf{PhysicalBody}$. It is comparable to the role played by blank nodes, at the class level. Existential restrictions are the core of the DLs $\mathcal{EL}$ (which stands for "existential logic") and its successor $\mathcal{EL}^{++}$ [2, 3]. Definition 3 subsumes both logics in terms of expressivity. There exist more expressive DLs featuring e.g. class complements ($\neg C$) and inverse properties (p^-) but the DL fragment being considered here has desirable computational properties, as we will see later.

The most expressive DL is denoted $\mathcal{SROIQ}$. Like any DL, it is known to be *decidable*: it is possible to resolve the satisfiability problem for any axiom α and any knowledge base $\mathcal{K}$ in a finite amount of time [99]. In contrast, the much more expressive FOL is undecidable, which is one of the reasons why DLs were chosen over FOL as the basis for OWL.

However, to keep decidability in $\mathcal{SROIQ}$, some further conditions apply to properties for a rule to be a valid DL rule. One of these conditions, called *regularity*, is the existence of a partial order between all properties in a knowledge base. These restrictions will not be developed here, the reader can safely assume they apply to all examples shown in this thesis. It is also worth noting that the notation introduced here differs from the classical DL notation with set-like operators ($\sqsubseteq, \sqcap, \sqcup$). The underlying objective is to remain consistent with the object notation that is otherwise develop in Ch. 4. This rule-based notation is inspired by earlier observations that all DL axioms can be expressed as rules of a certain form [39, 67]. The reader can refer to Krötsch's *Description Logic Rules* book for an exhaustive definition of DL rules [67].

3.2.2.2 *Semantics of Description Logics*

In order to decide whether $\mathcal{K} \models \alpha$ for some $\mathcal{K}$, α, the *semantics* of DL expressions must be defined. It is generally done in model-theoretical terms. Model theory relies on the notion of *interpretation*. Intuitively, every intelligent (WoT) agent builds an internal model of the physical world by mapping an object representation (or some assertions) to arbitrary symbols, i.e. by interpreting it. Formally, an interpretation $\mathcal{I}$ is composed of an (arbitrary) domain of interpretation denoted $\Delta^\mathcal{I}$ and an interpretation function denoted $\cdot^\mathcal{I}$ that maps entity names to $\Delta^\mathcal{I}$ [99]. More precisely, for a class name $C \in N^C$, we have $C^\mathcal{I} \subseteq \Delta^\mathcal{I}$, for a property name $p \in N^P$, we have $p^\mathcal{I} \subseteq \Delta^\mathcal{I} \times \Delta^\mathcal{I}$ and for an individual name $a \in N^I$, we have $a^\mathcal{I} \in \Delta^\mathcal{I}$.

Definition 4. *[69] Let f be a DL expression as per Def. 3. Let $\mathcal{I}$ be an interpretation defining a domain of interpretation $\Delta^\mathcal{I}$ and an interpretation function $\cdot^\mathcal{I}$. Let σ be a function $\sigma : V \cup N^I \mapsto \Delta^\mathcal{I}$ such that $\sigma(a) = a^\mathcal{I}$ for all $a \in N^I$ and σ satisfies the following constraints for f:*

- $\sigma(t) \in C^\mathcal{I}$

- *there exists* $\delta_i \in \Delta^{\mathcal{I}}$ *such that* $\langle \sigma(t), \delta_i \rangle \in p_i^{\mathcal{I}}$ *and* $\delta_i \in C_i^{\mathcal{I}}$ *for all* $i \in [1, k]$
- $\langle \sigma(t), \sigma(t'_j) \rangle \in p_j^{\mathcal{I}}$ *for all* $j \in [k+1, n]$

We say that $\mathcal{I}$ *satisfies* f *and write* $\mathcal{I} \models f$ *if such a function* σ *exists for* $\mathcal{I}$*. Moreover,* $\mathcal{I}$ *satisfies a rule* $H :- B$ *if* $\mathcal{I}$ *satisfies* H *or* $\mathcal{I}$ *fails to satisfy one of the formulas in* B*. A knowledge base* $\mathcal{K}$ *is satisfiable if there exists an interpretation that satisfies every rule in* $\mathcal{K}$*. We say that* $\mathcal{I}$ *is a model of* $\mathcal{K}$ *and write* $\mathcal{I} \models \mathcal{K}$*.*

The same notation ($\models$) is used for satisfiability and entailment, as we will see later when defining query answering. The two notions shall however not be confused.

Example 8. *In the previous example, one can observe that every model of* $\mathcal{K}_{ex}$ *will also satisfy the following rule:*

$$x\mathit{[hasProperty} \Rightarrow \mathit{Temperature]} :- x\mathit{:Space}.$$

Indeed, for a model $\mathcal{I}$ of $\mathcal{K}_{ex}$, if there is σ such that $\sigma(x) \in \mathsf{Space}^{\mathcal{I}} \subseteq \mathsf{Air}^{\mathcal{I}} \subseteq \mathsf{PhysicalBody}^{\mathcal{I}}$, then there must exist δ such that $\langle \sigma(x), \delta \rangle \in \mathsf{hasProperty}^{\mathcal{I}}$ and $\delta \in \mathsf{Temperature}^{\mathcal{I}}$.

3.2.2.3 *Relation to Other Logic Formalisms*

It is well-known that $\mathcal{SROIQ}$ is a syntactic variant of a strict subset of FOL. There exist other such subsets, especially in the field of database research where the most notable formalism is Datalog [20]. The interplay between DLs, FOL and Datalog has been explored in depth in the literature.

For instance, the intersection of $\mathcal{SROIQ}$ with Datalog is called DL Programs (DLP) [43]. This DL fragment presents the advantages of being supported by mature Datalog systems while also being *tractable*. That is, the satisfiability problem can be solved in polynomial time. DLP does not include existential restrictions, though. This limitation motivated the definition of $\mathcal{EL}$ Programs (ELP), which combines DLP with $\mathcal{EL}^{++}$ axioms, as well as some other Datalog constructs [69]. It is similar, yet more expressive than another logic allowing for existential reasoning over Datalog, called Datalog$^{\pm}$ [19].

Figure 8 shows the mutual inclusion of all these formalisms in terms of epxressiveness (taken to the most part from the foundation paper on Datalog$^{\pm}$ by Calí et al. [19]). The figure also includes tractability results for the problem of satisfiability under different logics. It has been proven that $\mathcal{EL}^{++}$ and ELP are tractable. Datalog$^{\pm}$ was also specifically designed to retain tractability. Sec. 3.3.1 includes a discussion on what formalism is suitable for existential reasoning in WoT use cases.

So far, we have only considered the problem of satisfiability, the simplest reasoning task. As already mentioned, reasoning is necessary but not sufficient to solve specific problems. Computational (WoT)

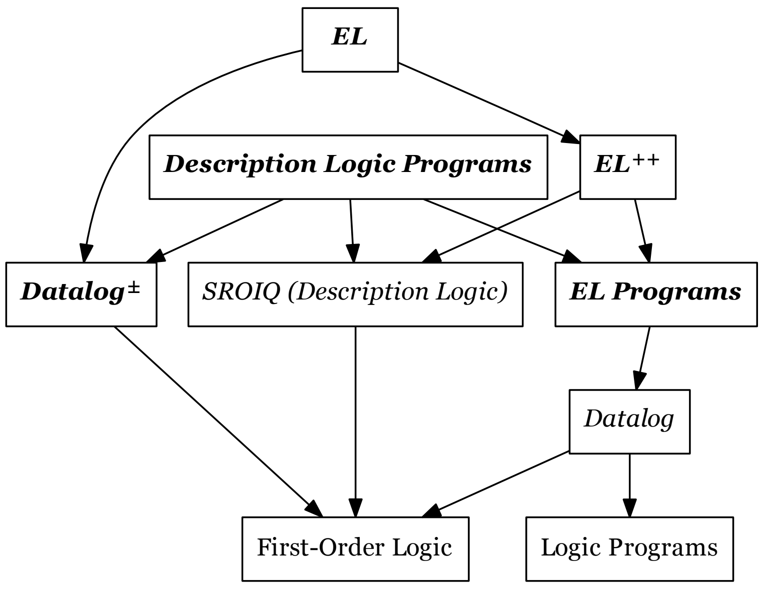


Figure 8: Partial order in terms of expressiveness between logic formalisms with existential restrictions related to DLs; tractable fragments are represented in bold font, decidable fragments in italic for the problem of satisfiability

agents likely need to query a database (e.g. to look at past observations) or extend a knowledge base with new observations. Next, we review two more advanced reasoning tasks: query answering and abduction.

3.2.3 *Query Answering and Abduction*

3.2.3.1 *Conjunctive Query Answering*

Query answering on knowledge bases with existential restrictions can be defined as follows:

Definition 5. *[68] A DL Conjunctive Query (CQ) $\mathcal{Q}$ is a conjunction of DL expressions without existential restrictions, that is, of formulas of the form*

$$\mathsf{t}{:}C[p_1 \to \mathsf{t}'_1, \ldots p_n \to \mathsf{t}'_n]$$

where $\mathsf{t}, \mathsf{t}'_1 \ldots \mathsf{t}'_n \in \mathsf{V} \cup \mathsf{N}^\mathsf{I}$, $p_1 \ldots p_n \in \mathsf{N}^\mathsf{P}$ and $C \in \mathsf{N}^\mathsf{C} \cup \{\top, \bot\}$. A knowledge base $\mathcal{K}$ entails $\mathcal{Q}$ if for all model $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I}$ also satisfies the expressions of $\mathcal{Q}$. The same notation is used for satisfiability and entailment, we write $\mathcal{K} \models \mathcal{Q}$.

Example 9. *In practice, CQ entailment is mostly relevant when a knowledge base includes ABox assertions. If we add the assertion $\alpha = 31.638{:}Space$ to $\mathcal{K}_{ex}$, then we have:*

$$\mathcal{K}_{ex} \cup \{\alpha\} \models 31.638[hasProperty \to \mathsf{y}{:}Temperature].$$

Example 9 introduces the notion of *substitution* for a CQ, which can also be formally defined. In the following definition and in the remainder of the paper, $\mathsf{N}^\mathsf{I}_\mathcal{K}$ denotes the set of individual names in $\mathcal{K}$ and $\mathrm{var}(\mathcal{Q})$, the set of variables in $\mathcal{Q}$.

Definition 6. *Let $\mathcal{K}$ be a knowledge base and $\mathcal{Q}$ be a CQ such that $\mathcal{K} \models \mathcal{Q}$. The CQ $\mathcal{Q}'$, obtained by replacing some $\mathsf{x} \in \mathsf{V}$ in $\mathcal{Q}$ with an individual name $\mathsf{a} \in \mathsf{N}^\mathsf{I}_\mathcal{K}$, is a substitution for $\mathcal{Q}$ if we also have $\mathcal{K} \models \mathcal{Q}'$. Moreover, $\mathcal{Q}'$ is a* minimal *substitution for $\mathcal{Q}$ if there is no other substitution $\mathcal{Q}''$ of $\mathcal{Q}$ such that $var(\mathcal{Q}'') \subseteq var(\mathcal{Q}')$ (up to a renaming of variables).*

It is easy to see that substitutions closely relate to the function σ defined in Def. 4. Indeed, given an model $\mathcal{I}$ of $\mathcal{K}$, a substitution exists for some $\mathsf{a} \in \mathsf{N}^\mathsf{I}_\mathcal{K}$ if and only if $\sigma(\mathsf{x}) = \mathsf{a}^\mathcal{I}$. We can further observe that if a substitution exists for some model $\mathcal{I}$, it is also valid for any other model $\mathcal{I}'$ of $\mathcal{K}$. *Query answering* is the problem of finding all minimal substitutions for a given CQ, which is a more general problem than query entailment.

A conjunctive query with no variable is called a Boolean CQ (BCQ). In practice, it is common to consider a CQ only as a set of BCQs obtained by substituting variables to named individuals. However, this conceptual shortcut excludes answers containing anonymous individuals, although they may also be semantically valid. The latter answers are precisely those of interest in WoT: the temperature property of room 31.638 is never explicitely asserted, as it would likely be in an actual TD document. For instance, in the CQ of Ex. 9, y is anonymous: there is no named entity that can be substituted to y so that the expression is still entailed by $\mathcal{K}_{\text{ex}}$.

One can note that query answering subsumes the problem of satisfiability. The example CQ that were just given can also be rewritten as the single axiom $x{:}\mathsf{Space}[\mathsf{hasProperty} \Rightarrow \mathsf{Temperature}]$. CQs are considered complex when they include at least one conjunction with shared variables between formulas [18]. It is comparable to queries with joins in relational databases. In fact, DL query answering shares many aspects with relational algebra. The DL subset that gained most attention with respect to query answering is called DL-Lite [17]. It has the property that queries can be rewritten into a single FOL expression that can be processed by relational databases (a property called *FOL-rewritability*). DL-Lite does not feature (qualified) existential restrictions, though.

In the general case, computational complexity for query answering depends on the size of both the query and the database. It is common to provide complexity results when either of them is bounded by a maximum size (see e.g. results for RDF stores by Guttierez et al. [48]). For fixed queries, it is called *data complexity* while for fixed databases, it is called *query complexity*. The former is more interesting in practice, since most queries are of much smaller size than databases[3]. For a DL fragment that excludes class disjunctions and complements, as well as transitive and functional properties [18], DL query answering is tractable for the data complexity. This fragment, which has much in common with Datalog$^{\pm}$, is subsumed by Def. 3. However, even for this DL fragment, there is no practical algorithm in the literature, as soon as knowledge bases include existential restrictions. It is also known that conjunctive query answering on $\mathcal{EL}^{++}$ is also tractable, which also features transitive properties [68]. Later in this chapter (Sec. 3.3.4), this result will be further extended, by introducing a tractable query answering algorithm for knowledge bases as defined in Def. 3.

To conclude this section on DL query answering, the closely related notion of *explanation* is introduced, mostly relevant in peer-to-peer (WoT) systems to guarantee that agents can entail similar results, given their own knowledge and some explanation provided by other agents.

3 in the context of knowledge bases, a third class of complexity, called *taxonomic complexity*, is generally of interest; it goes beyond the scope of this thesis, though, in which CQs are limited to ABox statements.

Definition 7. *[99] An explanation for an CQ $\mathcal{Q}$ is a knowledge base $\mathcal{E} \subseteq \mathcal{K}$ such that $\mathcal{E} \models \mathcal{Q}$ but for every subset $\mathcal{E}' \subset \mathcal{E}$, $\mathcal{E}' \not\models \mathcal{Q}$.*

The term 'explanation' is also used in the literature to refer to *abductive reasoning* procedures: if the definition above provides explanations for queries that are entailed by some knowledge base, abduction helps understand why some queries are *not* entailed. We review abduction methods for DLs in the following.

3.2.3.2 *Abductive Reasoning*

Abduction can be described in general terms as the procedure of finding missing axioms for a query to be entailed by a knowledge base $\mathcal{K}$. In that sense, every query answering problem is conceptually equivalent to an abduction problem, such that all "abduced" axiom is asserted in $\mathcal{K}$ [62]. However, there are infinitely many such axioms and concrete formalizations usually involve external information not directly provided by $\mathcal{K}$. For instance, solutions may only contain a subset of the properties, classes and individuals in $\mathcal{K}$, defined on a per application basis [119]. Early formalisms based on logic programming also involve *integrity constraints* allowing or not certain axioms [62]. In that respect, abductive reasoning significantly differs from satisfiability and query answering, both referred to as *deductive* reasoning tasks.

For DL queries, abduction is formally defined as follows:

Definition 8. *[119] Abduction consists in finding a knowledge base $\mathcal{K}'$ such that for a knowledge base $\mathcal{K}$ and a BCQ $\mathcal{Q}$, it holds that:*

- $\mathcal{K} \cup \mathcal{K}' \models \mathcal{Q}$
- $\mathcal{K} \cup \mathcal{K}' \not\models x{:}\bot (x \in V)$
- $\mathcal{K}' \not\models \mathcal{Q}$

Conditions 2 and 3 exclude trivial solutions. Abduction on general CQs can be answered by instantiating the CQ in every possible way using terms from N^I and a finite set of "fresh" IRIs (in order to obtain BCQs) and then applying Def. 8.

There exists various works addressing abduction on different DL fragments, with focus on tractability [29, 66, 119]. All of these works reduce the solution space to ABox axioms, which are the most relevant in practice. Complexity results are similar to those for query answering: abduction procedures exist for FOL-rewritable DLs and $\mathcal{EL}^{++}$. To the best of my knowledge, there is no algorithm covering Def. 3.

As we saw in our review for query answering, today's deductive reasonning engines tend to ignore existential restrictions. One possible strategy to overcome this limitation is to formulate query answering as an abduction problem, such that only property and class names involved in existential restrictions are abducible. It is equivalent to dynamically generating Skolem constants during query processing. This is e.g. the idea behind a recent experiment on Datalog$^{\pm}$ knowledge bases where query answering is implemented with an Abductive Logic Programming (ALP) engine [40]. Results of this experiment suggest that this approach performs well on large ABoxes.

In an earlier publication, I made an attempt to formalize and implement reasoning with TD documents using ALP [25]. The underlying DL was $\mathcal{EL}^{++}$. However, as we will see in Sec. 3.3.1, the expressivity needed in WoT goes beyond $\mathcal{EL}^{++}$ and yet, there is no state-of-the-art algorithm for more expressive DLs. In summary, it appears that existential reasoning procedures relevant for WoT are at the limit of what is known in DL research: new contributions are to be made to the state-of-the-art.

The following section presents the problem of WoT semantic discovery, formulated as an existential reasoning problem. It is formulated only in terms of query answering; possible abduction-based optimizations are left as future work.

3.3 A FRAMEWORK FOR SEMANTIC DISCOVERY ON THE WEB OF THINGS

We define the problem of semantic discovery as the discovery of possible interactions in a WoT system given a knowledge base that includes TD assertions. Semantic discovery accepts as input a set of TD documents and domain-specific knowledge provided by Web ontologies. We have already seen numerous ontologies in Ch. 2 without considering, however, their completeness in terms of axiomatization to solve problems like semantic discovery. We first review axioms potentially missing in the WoT ontology cloud (Sec. 3.3.1), before moving on to a formalization of semantic discovery (Sec. 3.3.2) and then to implementation considerations (Sec. 3.3.4).

3.3.1 *Expressiveness of a Web of Things Knowledge Base*

The focus of this chapter is the discovery of new relations between well identified 'things' by means of reasoning. Its basic assumption is that Web ontologies provide rules that can be exploited to assert these new relations in a knowledge graph, that is, a set of ABox assertions from TD documents. The rules of interest in the present context are those whose head includes existential restrictions (to introduce new vertices in the knowledge graph) or relations between one or two anonymous individuals (new edges). First, various DL features are being reviewed as to whether they refer to either of these aspects in order to decide on what DL fragment to target in WoT. The review

DL FEATURE	$\exists$	OCCURRENCES
Existential restriction	yes	562
Universal restriction	no	2,383
Min. cardinality constraint	yes	25
Max. cardinality constraint	no	7
Class complement	no	0
Self	yes	0
Transitive property	yes	8
Functional property	no	1,462
Inverse property	yes	280
Symmetric property	yes	11
Reflexive property	yes	0
Property chain	yes	6

Table 8: DL features for classes (top) and properties (bottom) and their contribution to existential reasoning; occurrences in the WoT ontology cloud are given for each feature

is followed by examples of such axioms with classes from the WoT ontology cloud.

Table 8 gives a list of the DL features that have an equivalent construct in OWL. The reader can refer to Appendix A for a mapping with DL rules as defined in Def. 3. The table is divided in two parts depending on whether they refer to classes (and to inferring new vertices) or properties (to infer new edges). First, it is easy to see that universal restrictions do not contribute to existential reasoning: for some class C, an interpretation $\mathcal{I}$ can satisfy restrictionson $C^{\mathcal{I}}$ regardless of whether $C^{\mathcal{I}}$ is empty or not. The observation also holds for class complements and maximum cardinality constraints (for cardinalities of two or more). On the other hand, minimum cardinality constraints are relevant since they subsume existential restrictions, the only dedicated construct to express existence. The last class feature, to allow individuals to refer to themselves, contribute to existential reasoning via complex property chains.

All DL features referring to properties are relevant in WoT except functional properties. By definition, the latter are used to identify entities, like primary keys in a database. Functionality implies a maximum cardinality of one and like universal restrictions and class complements, it does not help infer new individuals or relations. In contrast, property chains are the main construct to infer more knowledge about anonymous individuals, beside their mere existence. Therefore, they are important in WoT. Transitivity is a special case of property chain. Reflexive and inverse properties do not directly contribute to inferring assertions on anonymous individuals but they do simplify the axiomatization of Web ontologies, especially in defining property chains. We will discuss their usefulness later in this section. The same holds for symmetric properties, a special case of inverse properties.

Table 8 also shows occurrences of the different features in the WoT ontology cloud. It appears that most logical axioms are not relevant for existential reasoning. Still, it is possible to express rather simple rules with the vocabulary it exposes (the class, property and individual names). Examples of such rules can be provided in particular on two kinds of physical world entities: properties of features of interest (like a temperature) and objects related to features of interest from TD documents (like walls in a room).

Regarding properties (of features of interest), it is rather straightforward to create a knowledge base that gives the properties of physical world objects depending on their type. Fig. 9 provides an example of classification for physical bodies that extends OM, the ontology of units of measure. Physical bodies are first categorized according to their phase (liquid, solid or gas). Every physical body has an `om:Volume` and an `om:Temperature` but only fluids (liquid, gas) have an `om:Volumetric_flow_rate`. The first example of last section (Ex. 7) was a fragment of these axioms.

Physical quantities are further categorized according to the domain of physics that defines them. For instance, volume is relevant in geometry, property in thermodynamic and flow rates in mechanics. These categories help express relations between distinct objects because they share some properties.

Example 10. *The following rule states for instance that two intersecting fluids share the same thermodynamic properties:*

$$\begin{aligned} x_1\textit{[hasProperty} \rightarrow y\textit{]} :- \; & x_1\textit{:Fluid[intersects} \rightarrow x_2\textit{] and} \\ & x_2\textit{:Fluid[hasProperty} \rightarrow y\textit{] and} \\ & y\textit{:ThermodynamicProperty.} \end{aligned}$$

Finally, certain properties like the deformability of fluids can be expressed in terms of sensing and actuation on them. Deformability means that any mechanical action on a fluid implies changes of its geometry.

Example 11. *The following rule indirectly defines the deformability of fluids:*

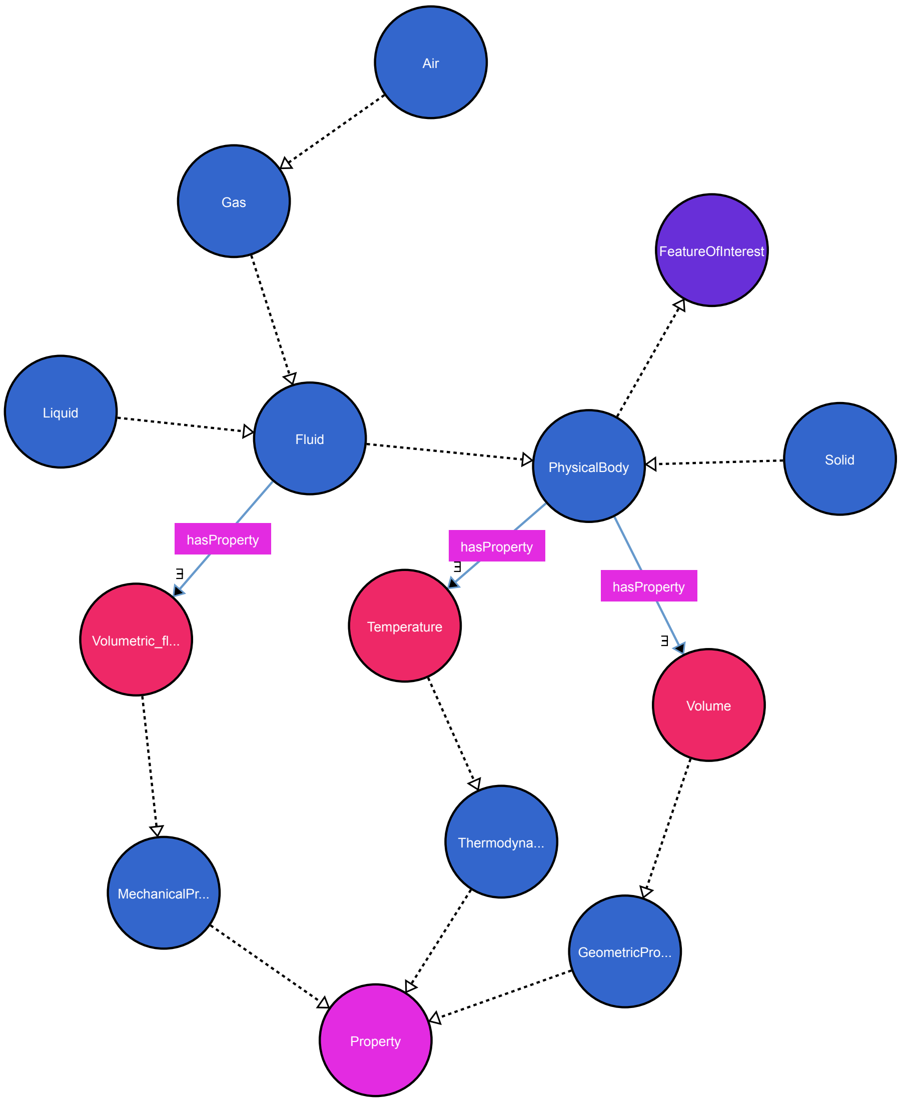


Figure 9: Main concepts of a knowledge base for physical bodies and their properties (VOWL notation); label colors encode namespaces (blue → example namespace, red → OM, purple → SSN)

$$\begin{aligned}x\textit{[actsOnProperty} \rightarrow z_2\textit{]} :- {} & x\textit{:Actuator[actsOnProperty} \rightarrow z_1\textit{] and}\\ & y\textit{:Fluid[}\\ & \quad \textit{hasProperty} \rightarrow z_1,\\ & \quad \textit{hasProperty} \rightarrow z_2\\ & \textit{] and}\\ & z_1\textit{:MechanicalProperty and}\\ & z_2\textit{:GeometricProperty.}\end{aligned}$$

Regarding physical world objects, the most interesting axioms to formalize are the geometric relations between physical bodies.

Example 12. *An example can be found in BOT, where* `bot:containsZone` *property is declared as transitive:*

$$\begin{aligned}x\textit{[containsZone} \rightarrow z\textit{]} :- {} & x\textit{[containsZone} \rightarrow y\textit{] and}\\ & y\textit{[containsZone} \rightarrow z\textit{].}\end{aligned}$$

It is possible to have even more generic relations considering only the phase of physical bodies.

Example 13. *For instance, the following rule states that a solid and a fluid that are both within some physical body necessarily intersect:*

$$\begin{aligned}x\textit{[intersects} \rightarrow z\textit{]} :- {} & x\textit{:Solid[within}\rightarrow y\textit{] and}\\ & y\textit{:PhysicalBody[contains} \rightarrow z\textit{] and}\\ & z\textit{:Fluid.}\end{aligned}$$

Geo-spatial relations in Ex. 13 are provided by schema.org (`schema:geospatiallyWithin` and `schema:geospatiallyContains`).

The DL feature from Table 8 that were used to express these DL rules are property chains and (indirectly) inverse and reflexive properties. The reader can refer to Appendix A for an explanation on how to transform DL rules to so-called "qualified" property chains. However, explicit inverse properties come with undesired effects: despite the obvious symmetry between `schema:geospatiallyWithin` and `schema:geospatiallyContains`, introducing an explicit symmetry relation between them would significantly increase the time complexity of query answering, as we will see later. Since inverse properties do not directly contribute to existential reasoning, they were purposely left out in the present formalization of DL knowledge bases (recall Def. 3). Experimentally, I have never been confronted with the case where they were strictly mandatory (contrary to property chains, which proved crucial). It is unclear, however, if this can be made a general rule.

Next, a WoT-specific problem is being formalized on the basis of existential reasoning, on the rules that were just presented. The resulting formalization holds regardless of the DL features allowed or not in a knowledge base. The choice that has been made in that respect is only relevant when considering practical implementation (Sec. 3.3.4).

3.3.2 *Problem Statement*

As mentioned previously, the basic assumption is that every WoT servient exposes a TD document containing logical assertions that use the vocabulary of some ontology, in which domain-specific rules are also defined. For simplicity and without loss of generality, it is assumed that all TD documents refer to the same ontology. Formally, a set of n servients expose each an ABox $\mathcal{A}_i, i \in [1, n]$ that uses the vocabulary of a shared CBox $\mathcal{C}$. We denote $\mathcal{A}$ the ABox $\bigcup_i \mathcal{A}_i$ and $\mathcal{K}$ the knowledge base defined as $\mathcal{A} \cup \mathcal{C}$.

On this basis, one can define a query answering problem to find paths between instances of the class Thing (including paths with anonymous individuals) such that every discovered path corresponds to a possible interaction between the servients that expose these Things. The latter will be referred to as *semantic discovery*, defined as follows.

Definition 9. *WoT semantic discovery on a knowledge base* $\mathcal{K} = \bigcup_i \mathcal{A}_i \cup \mathcal{C}$ *and for a CQ* $\mathcal{Q}$ *is the task of finding every minimal substitution* $\mathcal{Q}'$ *that includes* $\mathsf{a} \in \mathsf{N}^I_{\mathcal{A}_i}$ *and* $\mathsf{b} \in \mathsf{N}^I_{\mathcal{A}_j}$ *distinct such that* $\mathcal{K} \models \mathsf{a}\mathord{:}\mathsf{Thing}$ *and* $\mathsf{b}\mathord{:}\mathsf{Thing}$.

Interactions of particular interest are those between two servients observing or acting on the same property of some feature of interest.

Example 14. *Let us define some minimal TD for a radiator* ($\mathcal{A}_1$)*:*

$$\mathsf{31.638{:}Space[contains \rightarrow s1{:}Radiator]}.$$

as well as a TD for a temperature sensor ($\mathcal{A}_2$)*:*

$$\mathsf{31.638{:}Space[contains \rightarrow s2{:}TemperatureSensor]}.$$

and a CBox similar to what was presented in Sec. 3.3.1 ($\mathcal{C}$)*:*

$$
\begin{aligned}
x{:}[\mathit{actsOnProperty} \Rightarrow \mathit{Temperature}] &:- x{:}\mathit{Radiator}.\\
x{:}[\mathit{observes} \Rightarrow \mathit{Temperature}] &:- x{:}\mathit{TemperatureSensor}.\\
x{:}[\mathit{hasProperty} \Rightarrow \mathit{Temperature}] &:- x{:}\mathit{Space}.\\
x[\mathit{actsOnProperty} \rightarrow z] &:- x[\\
&\quad \mathit{contains} \rightarrow y,\\
&\quad \mathit{hasProperty} \rightarrow z{:}\mathit{Temperature}\\
&]\ \mathit{and}\\
&y[\mathit{actsOnProperty} \Rightarrow \mathit{Temperature}].\\
x[\mathit{observes} \rightarrow z] &:- x[\\
&\quad \mathit{contains} \rightarrow y,\\
&\quad \mathit{hasProperty} \rightarrow z{:}\mathit{Temperature}\\
&]\ \mathit{and}\\
&y[\mathit{observes} \Rightarrow \mathit{Temperature}].
\end{aligned}
$$

Let $\mathcal{K}$ be the knowledge base $\mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{C}$ and $\mathcal{Q}$ the following CQ:

$$x[\mathit{observes} \rightarrow z]\ \mathit{and}\ y[\mathit{actsOnProperty} \rightarrow z].$$

The output of semantic discovery on $\mathcal{K}$ for $\mathcal{Q}$ is the single minimal substitution $\mathcal{Q}'$:

$$\mathit{s1}[\mathit{observes} \rightarrow x]\ \mathit{and}\ \mathit{s2}[\mathit{actsOnProperty} \rightarrow x].$$

In practice, for a broader discovery taking into account sensor/sensor and actuator/actuator interactions, one can introduce the property relatesToProperty, as follows:

$$
\begin{aligned}
x[\mathsf{relatesToProperty} \rightarrow y] &:- x[\mathsf{observes} \rightarrow y].\\
x[\mathsf{relatesToProperty} \rightarrow y] &:- x[\mathsf{actsOnProperty} \rightarrow y].
\end{aligned}
$$

and then run semantic discovery on the following query:

$$x[\mathsf{relatesToProperty} \rightarrow z]\ \mathsf{and}\ y[\mathsf{relatesToProperty} \rightarrow z].$$

This is the query that was used for experiments, presented later (Sec. 3.4).

When an interaction indeed takes place in a system, it can be stated by an `ssn:hasSubSystem` relation between a composite system and its sub-systems. The assumption is that TD documents are managed by the servients themselves, as opposed to being stored in a central RDF store. As a consequence, these TD documents may be updated with the results of semantic discovery, in order for all servients involved in a new interactions to have sufficient knowledge about it. That is, servients should be able to entail the discovered CQs from their own ABox. This relates to the notion of query explanation we defined earlier in this chapter (Def. 7).

Definition 10. *Let $\mathcal{E}$ be an explanation for a CQ $\mathcal{Q}$ on $\mathcal{K}$. The update ABox for $\mathcal{Q}$ is the ABox $\mathcal{A}' = \bigcup_i \mathcal{A}'_i$ such that for all $\mathcal{A}_i$ where $\mathsf{N}^{\mathrm{I}}_{\mathcal{A}_i} \cap \mathsf{N}^{\mathrm{I}}_{\mathcal{E}} \neq \emptyset$, $\mathcal{A}'_i = \mathcal{E} \setminus \mathcal{A}_i$.*

Given Def. 10, it then holds that $(\mathcal{A}_i \cup \mathcal{A}'_i) \cup \mathcal{C} \models \mathcal{Q}$. One can further note that update ABoxes can be computed incrementally. That is, if $\mathcal{A}'_i$ is the update ABox resulting from a first discovery on servient with ABox $\mathcal{A}_i$, when discovery is performed a second time, the new update ABox $\mathcal{A}''_i$ can be computed against $\mathcal{A}_i \cup \mathcal{A}'_i$ such that $\mathcal{A}'_i \cap \mathcal{A}''_i = \emptyset$. Incremental update of ABoxes has practical benefits since it is likely that servients enter a WoT system at different times over the lifecycle of that system.

Example 15. *The update ABox for Ex. 14 is the union of the following ABox for the radiator ($\mathcal{A}'_1$):*

31.638[contains → s2:TemperatureSensor].

and the following ABox for the temperature sensor ($\mathcal{A}'_2$):

31.638[contains → s1:Radiator].

In the general case, the update ABox for a given servient is inversely proportional to its prior level of knowledge.

3.3.3 *Generalization*

The semantic discovery framework that was just presented aims at discovering interactions between WoT servients. We can now come back to the initial abstraction for WoT systems of Sec. 1.2.3 and provide general semantics for WoT as a mapping between a graph of interactions and a knwoledge graph.

We recall that WoT were defined as systems as multi-agent systems, primarily defined by a graph of interactions. The latter is a graph $G = \langle V, E \rangle$ such that V is a set of servients and E a set of unordered pairs $\{x, y\}$ whenever servient x interacts with servient y. Considering the (possibly contradictory) goals of WoT servients is out of the scope of this thesis, as is an analysis of the events to which they react. Instead, WoT semantics only aim at preserving the logical consistency of the system by guaranteeing that every interaction can be *explained* in terms of knowledge graph. As a consequence, edges of an graph of interactions are undirected, since a client can initiate the request as well as it can receive an asynchronous notification from the server in a particular exchange. In fact, the interaction might even be *mediated* by some other device but it is of little interest from a semantic point of view to distinguish the case where interactions are mediated and when they are direct (they are then called *peer-to-peer* interactions).

In the previous section, it was assumed that every servient interaction relates to a specific CQ $\mathcal{Q}$ involving the Things they expose, such that for a knowledge base $\mathcal{K}$ describing the system and its environment, we have $\mathcal{K} \models \mathcal{Q}$. In other words, we assumed that every edge in G is labeled with some minimal substitution for $\mathcal{Q}$. If we generalize, edges of a graph of interactions can be labeled with any DL expression entailed by $\mathcal{K}$.

Definition 11. *Let* $G = \langle V, E \rangle$ *be a graph of interactions and* $\mathcal{K}$ *a knowledge base for a WoT system. Let* F *the set of DL formulas. The semantics of a WoT system is the bijection* $[\![\cdot]\!]_{\mathcal{K}} : E \mapsto F$ *such that for* $e \in E$ *and* $f = [\![e]\!]_{\mathcal{K}} \in F$*:*

- $\mathcal{K} \models f$
- $\mathcal{K} \models$ *a:Thing and b:Thing for* $a, b \in N^I$ *included in* f
- f *is minimal as per Def. 6*

Given Def. 11, several theoretical problems can be defined for a WoT system. We can e.g. reformulate semantic discovery as the problem of finding the inverse of $[\![\cdot]\!]_{\mathcal{K}}$ for some CQ $\mathcal{Q}$, i.e. for a subset of F. Another problem of interest is that of computing $[\![e]\!]_{\mathcal{K}}$ for every e in some graph of interactions G.

Example 16. *Consider the knowledge base* $\mathcal{K}$ *of Ex. 14 such that the two servients s1, s2 are synchronized to keep the temperature of room 31.638 constant. We then have* $G = \langle \{s1, s2\}, \{e = \{s1, s2\}\} \rangle$ *and:*

$$[\![e]\!]_{\mathcal{K}} = \textit{s1[observes} \rightarrow x] \text{ and } \textit{s2[actsOnProperty} \rightarrow x].$$

The semantics introduced here—the corner stone of this thesis—closely relates to DL query answering, regardless of the problem we consider. Next, we shall consider concrete implementation aspects of query answering in the presence of existential restrictions.

3.3.4 *A Tractable Approach*

In Ex. 14, discovery is only successful if existential restrictions in $\mathcal{C}$ are correctly processed. Indeed, no temperature property of room 31.638 is explicitely named, although it is known that every room has some temperature (as has every body of air). As underlined in the earlier review of the state-of-the-art in DL reasoning (Sec. 3.2), there is no dedicated algorithm for conjunctive query answering with existential restrictions: most known implementations of DL reasoners that provide query answering are not complete. A straightforward remedy is to *emulate* the original knowledge base by creating Skolem constants wherever the existence of some entity is stated and run classical query answering algorithms over the emulated knowledge base. The SKOLEMIZE procedure introduced in the following follows that principle (Alg. 1). Equivalence in terms of query answering between a knowledge base and its emulated form is then proven (Th. 2).

Algorithm 1 Skolemization algorithm for a knowledge base $\mathcal{K}$ with existential retrictions. S is a subset of the individual names in $\mathcal{K}$ and n an arbitrary integer.

```
1:  function SKOLEMIZE($\mathcal{K}$, $S \subseteq N^I_{\mathcal{K}}$, $n$)
2:      Let $\mathcal{K}' = \mathcal{K}$, $S' = \emptyset$
3:      for all $a \in S, p \in N^P_{\mathcal{K}}, C \in N^C_{\mathcal{K}}$, s.t. $\mathcal{K} \models a[p \Rightarrow C]$. do
4:          Let $b$ be a fresh individual (Skolem constant)
5:          $\mathcal{K}' := \mathcal{K}' \cup \{a[p \rightarrow b{:}C].\}$
6:          $S' := S' \cup \{b\}$
7:      end for
8:      if $n = 1$ then
9:          for all $f$ in $\mathcal{K}$ of the form $t[p \Rightarrow C]$ do
10:             Let $x$ be a fresh variable, $f' := t[p \rightarrow x{:}C]$
11:             Replace all occurence of $f$ in $\mathcal{K}'$ with $f'$
12:         end for
13:         return $\mathcal{K}'$
14:     else
15:         return SKOLEMIZE($\mathcal{K}'$, $S'$, $n - 1$)
16:     end if
17: end function
```

To prove the equivalence between an input $\mathcal{K}$ and the output of Alg. 1, it must first be proven that DL bases as defined in Def. 3 have the so-called *finite model property* [99], which would guarantee the termination of SKOLEMIZE. Theorem 1 puts it in formal terms.

Theorem 1. *Let $\mathcal{K}$ be a DL knowledge base. $\mathcal{K}$ is satisfiable if and only if there exists a finite model $\mathcal{I}$ of $\mathcal{K}$.*

Proof. The theorem immediately follows from the fact that the definition of DL rules given in the thesis excludes cardinality constraints [4]. □

The equivalence provided by Alg. 1 can now be proven.

Theorem 2. *Let $\mathcal{K}$ be a DL knowledge base, i.e.a set of rules of the form $H :- B$. Let n be the maximum number of variables in B over $\mathcal{K}$ and $\mathcal{Q}$ be a conjunctive query using the vocabulary of $\mathcal{K}$. The knowledge base $\mathcal{K}_n$, obtained by the application of* SKOLEMIZE *(Alg. 1) on $\mathcal{K}$, $N^I_{\mathcal{K}}$ and n, emulates $\mathcal{K}$ and $\mathcal{K} \models \mathcal{Q}$ if and only if $\mathcal{K}_n \models \mathcal{Q}$.*

Proof. We start by observing that for all model $\mathcal{I}$ of $\mathcal{K}$, all path $\langle a^{\mathcal{I}}, \delta_1 \rangle$, $\dots \langle \delta_{k-1}, \delta_k \rangle$ is at most of length n, for some $a \in N^I_{\mathcal{K}}$ and $\delta_1, \dots, \delta_k$ anonymous. Proof: for all rule $H :- B$ in $\mathcal{K}$, s.t. $\mathcal{I}$ satisfies both H and B, all property chains in B are of the form $t_1[p_1 \rightarrow t_2], \dots, t_{n-1}[p_n \rightarrow t_n]$ (as per Def. 3). The resulting path in $\Delta^{\mathcal{I}}$ is of maximum length when $t_i^{\mathcal{I}} \neq t_j^{\mathcal{I}}$ for all distinct $i, j \in [1, n]$, that is, of length n.

We can now proceed to a proof for emulation, which is defined by two criteria [99]:

(1) Hypothesis: $\mathcal{I} \models \mathcal{K}_n$. The goal is to prove that $\mathcal{I}$ is also a model of $\mathcal{K}$. If, for all formula in $\mathcal{K}_n$, there is σ satisfying Def. 4 for $\mathcal{I}$, then it also satisfies all formulas of $\mathcal{K}$ with no existential restriction. If a formula includes a restriction $t_i[p \Rightarrow C]$ in $\mathcal{K}$, then there is an equivalent formula $t_i[p \rightarrow x{:}C]$ in $\mathcal{K}_n$, s.t. $\langle \sigma(t_i), \sigma(x) \rangle \in p^{\mathcal{I}}$ and $\sigma(x) \in C^{\mathcal{I}}$. Then $\delta_i = \sigma(x)$. If $\mathcal{I}$ fails at satisfying some formula in a rule body, then no σ exists and more precisely, if it fails for some existential restriction, there is no domain element $\sigma(x)$ and therefore no δ_i either. In both cases, $\mathcal{I} \models \mathcal{K}$.

(2) Hypothesis: $\mathcal{I} \models \mathcal{K}$. The goal is now to construct $\mathcal{I}'$, s.t. $\mathcal{I}' \models \mathcal{K}_n$, $\Delta^{\mathcal{I}'} = \Delta^{\mathcal{I}}$ and $\cdot^{\mathcal{I}'} = \cdot^{\mathcal{I}}$ for every name in $N^C_{\mathcal{K}} \cup N^P_{\mathcal{K}} \cup N^I_{\mathcal{K}}$. We define $\mathcal{I}'$, s.t. the latter constraint is met. Let σ be the function satisfying all constraints on $\mathcal{K}$ for $\mathcal{I}$, s.t. there is no map $\mu : \Delta^{\mathcal{I}} \mapsto \Delta^{\mathcal{I}}$, s.t. $\sigma \circ \mu$ also satisfies all constraints on $\mathcal{K}$. σ always exists and is unique (up to a renaming of individuals) [48]. If $\sigma(x)$ is anonymous for some variable x in $\mathcal{K}$, then there is a path $\langle a^{\mathcal{I}}, \delta_1 \rangle$, $\dots$, $\langle \delta_{k-1}, \delta_k \rangle$ with $k \leqslant n$ (as per our preliminary observation), $a^{\mathcal{I}}$ some named individual and $\delta_k = \sigma(t)$. Furthermore, let this path be the shortest path, s.t. for some $p_1, \dots, p_k$, we have $\langle a^{\mathcal{I}}, \delta_1 \rangle \in p_1^{\mathcal{I}}$, $\dots$, $\langle \delta_{k-1}^{\mathcal{I}}, \delta_k \rangle \in p_k^{\mathcal{I}}$ and for some $C_1, \dots, C_k$, we have $\delta_1 \in C_1^{\mathcal{I}}$, $\dots$, $\delta_k \in C_k^{\mathcal{I}}$, in order for this path to be unique (otherwise, it would be possible to define μ). It follows that $a[p_1 \Rightarrow C_1]$ and therefore, there must also be some Skolem constant b_1, s.t. $a[p_1 \rightarrow b_1{:}C_1]$ is in $\mathcal{K}_n$. We then define $b_1^{\mathcal{I}'} = \delta_1$, from which follows that $\langle a^{\mathcal{I}'}, b_1^{\mathcal{I}'} \rangle \in p^{\mathcal{I}'}$ and $b_1^{\mathcal{I}'} \in C^{\mathcal{I}'}$, given that $p^{\mathcal{I}'} = p^{\mathcal{I}}$ and $C^{\mathcal{I}'} = C^{\mathcal{I}}$. Therefore, $\mathcal{I}' \models a[p \rightarrow b_1{:}C]$. Similarly, we define $b_i^{\mathcal{I}'} = \delta_i$ for all $i \in [2, k]$, s.t. $\mathcal{I}' \models b_{i-1}[p_{i-1} \rightarrow b_i{:}C_i]$, generated in the i-th recursive call of SKOLEMIZE ($i \leqslant k \leqslant n$). Finally, we have $\mathcal{I}' \models \mathcal{K}_n$ with σ satisfying all constraints on $\mathcal{K}_n$.

We then proceed to a proof for reciprocity w.r.t query answering:

(if) Hypothesis: $\mathcal{K}_n \models \mathcal{Q}$. Because $\mathcal{K}_n$ emulates $\mathcal{K}$, all model $\mathcal{I}$ of $\mathcal{K}$ has the same image as some model $\mathcal{I}'$ of $\mathcal{K}_n$ (and $\mathcal{Q}$) using some function σ'. The goal is to prove that σ' also satisfies constraints on $\mathcal{Q}$ for $\mathcal{I}$. By virtue of emulation, we have $C^{\mathcal{I}'} = C^{\mathcal{I}}$ for all $C \in N^C_{\mathcal{K}}$, from which directly follows that for all $t{:}C$ in $\mathcal{Q}$, $\sigma'(t) \in C^{\mathcal{I}}$. Similarly, $p^{\mathcal{I}'} = p^{\mathcal{I}}$ for all $p \in N^P_{\mathcal{K}}$ and $\langle \sigma'(t_1), \sigma'(t_2) \rangle \in p^{\mathcal{I}}$ for all

$t_1[p \rightarrow t_2]$ in $\mathcal{Q}$. Therefore, σ' indeed satisfies all constraints for $\mathcal{I}$.

(only if) Hypothesis: $\mathcal{K} \models \mathcal{Q}$. Do we have $\mathcal{I}' \models \mathcal{Q}$ for all model of $\mathcal{K}_n$? We proceed by contradiction and assume $\mathcal{I}' \not\models \mathcal{Q}$ for some model $\mathcal{I}'$ of $\mathcal{K}_n$. It still holds that $\mathcal{I}' \models \mathcal{K}_n$, therefore $\mathcal{I}' \models \mathcal{K}$ (by virtue of emulation) and $\mathcal{I}' \models \mathcal{Q}$ (Def. 5), which contradicts the hypothesis.

□

Alg. 1 requires reasoning only to satisfy simple existential restrictions on individuals, that is, expressions of the form $a[p \Rightarrow C]$ (l. 3). The latter is not a complex query and, in fact, it is not even a BCQ (Def. 5 excludes existential restrictions). Proving that $\mathcal{K} \models a[p \Rightarrow C]$, what is called *instance checking*, is a much simpler task than query answering. The DL literature provides an instance checking algorithm for any DL fragment. In particular, a tractable algorithm based on some transformation to Datalog exists for ELP (a "combination" of $\mathcal{EL}^{++}$ and DL programs). Intuitively, the SKOLEMIZE procedure reduces general query answering to a finite number of applications of instance checking. The increase in size during skolemization is bounded by $(N^I_{\mathcal{K}}.N^R_{\mathcal{K}}.N^C_{\mathcal{K}})^n$ in the worst case (max. n variables per rule). For arbitrary knowledge bases, this leads to an exponential blow-up. In practice, however, it is reasonable to assume that n is bounded. For instance, in all experiments, rules in $\mathcal{K}$ have at most five variables. This allows one to retain tractability, as stated in the following theorem.

Theorem 3. *Let $\mathcal{K}$ be a knowledge base as per Def. 3 such that for all rule $H :- B$ in $\mathcal{K}$, B has at most n variables. Query answering for $\mathcal{K}$ (with existential reasoning) can be computed in polynomial time with respect to the size of $\mathcal{K}$.*

Proof. By definition, any DL knowledge base as defined in Def. 3 is in the DL fragment of ELP, for which satisfiability and classification are polynomial [69]. During the application of SKOLEMIZE, classification is called a finite amount of time on a knowledge base whose size (w.r.t. class, property and individual names) increases linearly w.r.t. the input $\mathcal{K}$ (for a fixed n). SKOLEMIZE therefore returns in a polynomial amount of time w.r.t. the size of $\mathcal{K}$.

Query answering on the output knowledge base $\mathcal{K}'$ is in turn polynomial w.r.t. the size of $\mathcal{K}'$, which immediately follows from the observation that it can be reduced to instance checking (in ELP), given that every individual for some model $\mathcal{I}'$ of $\mathcal{K}'$ is named. □

As already mentioned, all the logics that are considered in this chapter are standardized by OWL, the reference language for Web ontologies. There exists numerous commercial solutions for RDF storage that include standard OWL reasoning, like GraphDB[4] and Stardog[5]. The implementation of SKOLEMIZE against such RDF stores is rather straightforward. Query answering without existential restrictions, as output by SKOLEMIZE, can then be delegated to the SPARQL engine these solutions implement. SPARQL is the standard RDF query language [44].

OWL reasoners are designed for expressive DL axioms, most of which being out of the scope of this thesis to retain tractability. It is therefore not clear whether RDF stores will have satisfactory performances for WoT use case. An alternative consists in transforming DL rules into equivalent Datalog programs with specific constructs to then load them into deductive database systems like RDFox [88] or XSB[6] [102]. In contrast to RDF stores, however, most deductive database systems are prototypical or dedicated to research. The following section includes details on the implementation that was used as are provided experiments to evaluate the practical feasability of the approach.

3.4 PROOF OF CONCEPT & EVALUATION

The semantic discovery framework defined in theoretical terms in this chapter (Def. 9) relies on a discovery agent that first collects available TD documents in a system and then resolves the given query answering problem. These two steps must be performed regardless of the type of system (mediated or peer-to-peer). Optionally, for peer-to-peer systems, the individual components of the system—the servients—can be notified of the discovered relations with other servients. One can call this discovery agent, in simple terms, a TDir.

My TDir implementation offers a registration Web interface designed after the IETF Resource Directory specification[7] and backed by an RDF store. TD documents can be registered in plain JSON, one of the RDF serialization formats (Turtle, RDF/XML or JSON-LD) or the IETF CoRE Link format and its derivatives [109]. The latter target constrained environments by providing concise serializations (see next chapter for a more detailed discussion on the so-called Embedded Web). To a lesser extent, this TDir implementation is also inspired from the early Hypercat specification, a Web resource catalogue with RDF-like annotations[8], with the difference that this implementation also provides a SPARQL interface with off-the-shelf OWL reasoning. The source code of this TDir can be found online[9].

4 http://graphdb.ontotext.com/
5 https://www.stardog.com/
6 if used in its Datalog flavor, with tabling.
7 https://datatracker.ietf.org/doc/draft-ietf-core-resource-directory/
8 http://hypercat.io/
9 https://github.com/thingweb/thingweb-directory/

TDir-based semantic discovery was tested on two use cases to empirically evaluate its feasability, in different domains of application for WoT: BA, on a sensor network setup provided by Intel Labs, and Industrial Control Systems (ICS) with the water management use case that was presented in Sec. 1.2.2.

3.4.1 *Use Case: Intel Labs Sensor Network*

The Intel Labs sensor network is an experimental setup originally designed to study the routing of sensor measurements in a constrained node network[10]. It consists of 54 wireless devices called *motes* with as little as 8kB RAM, deployed homogeneously over an entire floor of the Intel Labs building to cover different zones: meeting rooms, closed offices, open space, entrance hall and kitchen (Fig. 10). All devices have identical capabilities: they can measure temperature, humidity and illuminance.

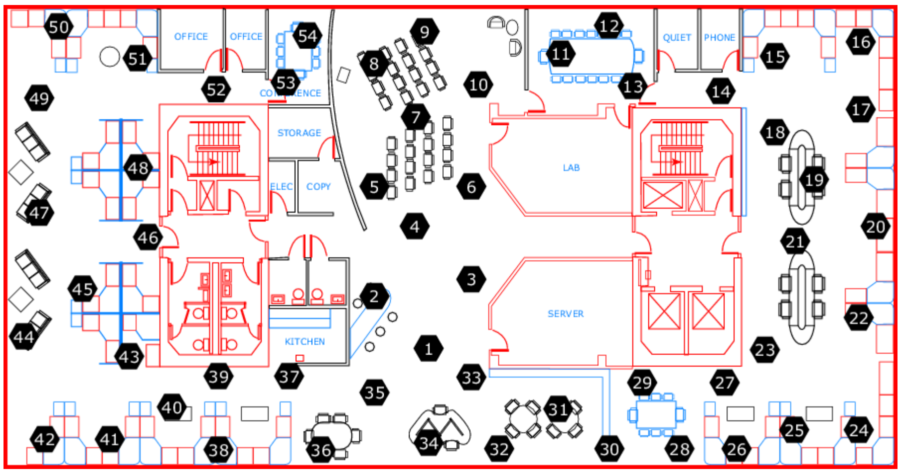


Figure 10: Map of the Intel Labs sensor network (Source: Intel Labs)

This setup was not originally intended for WoT use cases but it may corresponding to what a future WoT system will look like: a dense network of low-power connected devices that are capable of self-organzing in order to collectively achieve certain goals. In particular, the following system can be identified for this sensor network:

HVAC ANOMALY DETECTION Devices measuring temperature and humidity on the same body of air exchange measurements with their neighbors to detect anomalies, i.e. high localized gradient values, in Heating, Ventilation & Air Conditioning (HVAC) equipment.

If the system included actuators and sensors of different kinds, the range of possible interactions would be significantly wider. Yet, this

10 http://db.csail.mit.edu/labdata/labdata.html

particular use case offers a simple setup to test the feasibility of semantic discovery in practice. The only criterion to decide whether two wireless devices should interact is their location, that is, the zone they observe (in the sense of `bot:Zone`, a bounded space relevant for HVAC and lighting systems). Here, it is assumed that a map of the Intel Labs office giving its different zones is available in RDF and, *a fortiori*, as DL assertions to include to the knowledge base used for semantic discovery. It is safe to assume that for every BA use case, such information is available. Recall e.g. the IFC and Project Haystack information models, both ported to OWL (Table 3): these models are used by architects and civil engineers throughout the design of a building. They are part of what is commonly referred to as a Building Information Model (BIM).

Example 17. *Below is an excerpt of the BIM that was used for this experiment. First, a small class hierarchy was defined for the Intel Labs office.*

$$\begin{aligned} x{:}\mathit{Solid} &:- x{:}\mathit{Wall}. \\ x{:}\mathit{Air} &:- x{:}\mathit{Space}. \\ x{:}\mathit{Zone} &:- x{:}\mathit{Space}. \\ x{:}\mathit{Space} &:- x{:}\mathit{Hall}. \\ x{:}\mathit{Space} &:- x{:}\mathit{ConferenceRoom}. \end{aligned}$$

Then, the adjacency relations between spaces and separating walls were asserted, as follows (excerpt):

$$\begin{aligned} &\mathit{hall{:}Hall[adjacentElement} \to \mathit{leftWall{:}Wall]}. \\ &\mathit{room{:}ConferenceRoom[} \\ &\quad \mathit{adjacentElement} \to \mathit{leftWall{:}Wall}, \\ &\quad \mathit{adjacentElement} \to \mathit{rightWall{:}Wall} \\ &]. \end{aligned}$$

The rest of the knowledge base for this use case is similar to what was given in Ex. 14 (with instances of TemperatureSensor only). The complete RDF documents can be found online[11]. Semantic discovery ran for the CQ that was provided earlier (after introducing the generic property relatesToProperty), for which one obtains the graph of interactions shown on Fig. 11. What is immediately identifiable on this figure are the three complete subgraphs that dominate the distribution of edges. Each represent a portion of the open space (left side, hall and right side): all devices located in the same zone actually observe the same temperature value. They can therefore all interact

11 https://github.com/vcharpenay/urdf-store-exp

with any other device in that zone in order to detect anomalies on the measured temperature. Other subgraphs of size one, two and three represent isolated devices located in meeting rooms or closed offices.

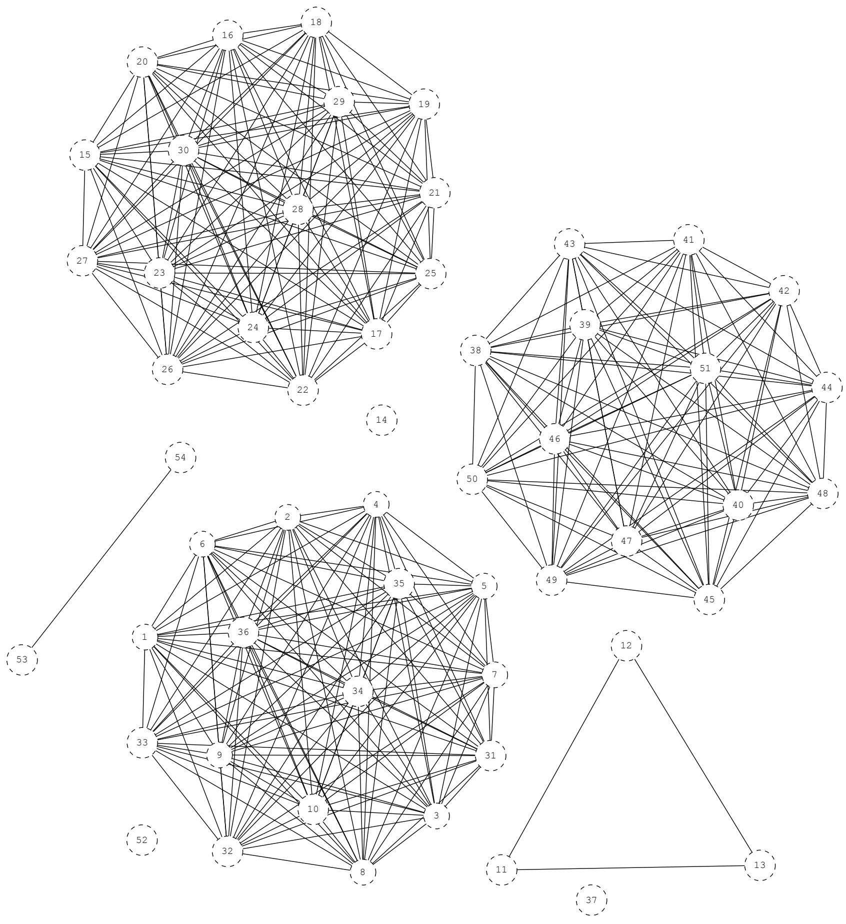


Figure 11: Graph of interactions for the Intel Labs sensor network

What this graph reveals is the high imbalance in terms of coverage of the physical space: if one of the isolated devices fails, a whole segment of the office will not be covered anymore, while the failure of a device in the open space would not affect the "physical" coverage of the sensor network. The Intel Labs dataset comes with temporal data indicating the connectivity status of devices between February 28th and April 5th, 2004. This data was plotted on Fig. 12. Over the course of the experiment, more and more devices go offline, most likely because of power outage, until the percentage of online devices (mote coverage) eventually reaches 0% on April 3rd. As a comparison, a graph of interactions was computed at every point in time and computed the percentage of remaining connected subgraphs compared to the original graph, that is, the percentage of anomaly detection systems one could still implement with the remaining devices. Surprisingly, we observe that the first devices to go offline are those placed in open spaces and thus redundant: until March 22nd all subgraphs are still present although 20% of devices went offline. Then, until April 1st, although only 10% of devices are online, we still have 50% of physical coverage.

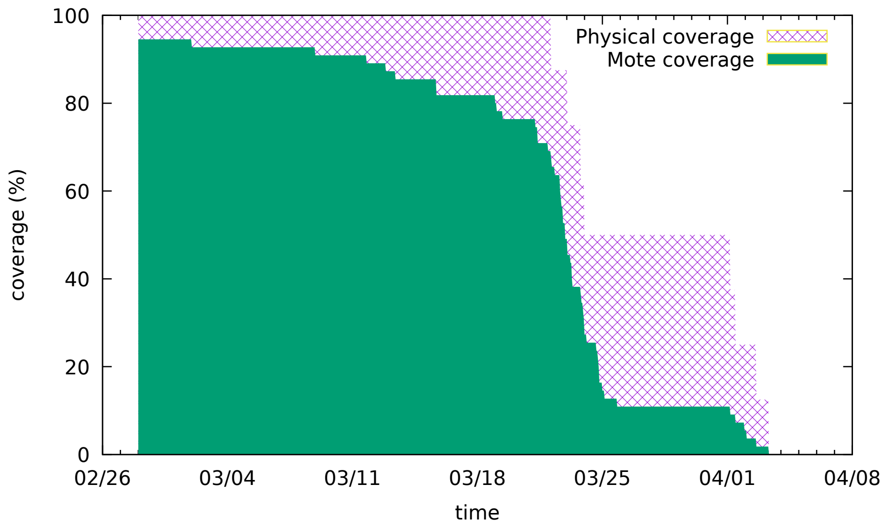
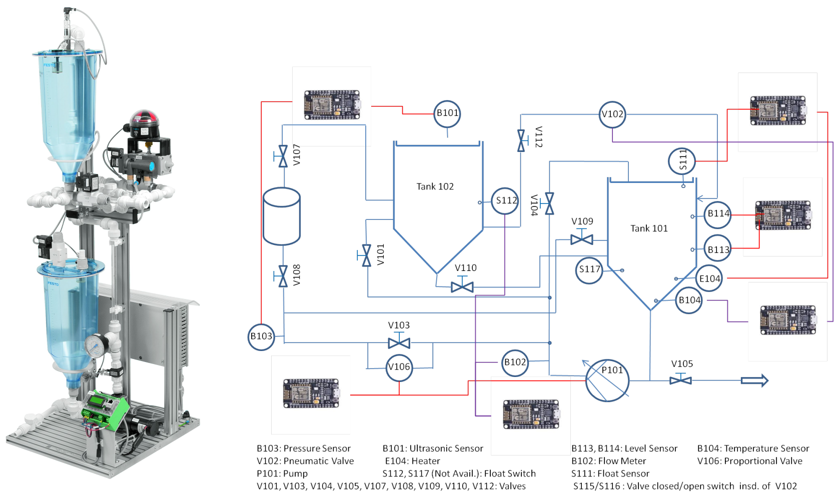


Figure 12: Evolution over time of mote coverage (number of online motes) and physical coverage (number of connected components in a graph of interactions) in the Intel Labs sensor network

3.4.2 *Use Case: Water Management Plant*

The Intel Labs setup was highly homogeneous and exploratory insofar as no existing WoT system shares the same characteristics yet. In contrast, the other setup that was experimented with is heterogeneous and the WoT servients it embeds are fully operational. It consists in an industrial workstation that includes water tanks and circulation pipes, equipped with various automation devices like valves, water level sensors, a flow meter, a temperature sensor, a pump, and a heater. This workstation is meant to simulate a water treatment plant in which water flows from one tank to the other. It has been equipped with six Micro-controller Units (MCUs) acting as WoT servients with IP connectivity. An overview of the workstation is provided in Fig. 13.

These micro-controllers are ESP8266 (64kB RAM, 80 MHz), they all embed a TD document in a lightweight RDF store designed for constrained devices, called the µRDF store, which is the main topic of Ch. 4. These TD documents include assertions that use SOSA, SSN and eCl@ss. eCl@ssOWL was slightly extended for the needs of the experiment. Essentially, the extension consists in an alignment with SSN



(a) Model (b) Logical circuit

Figure 13: Overview of a water management plant model with ESP8266 micro-controllers

and links between sensors and actuators and the physical quantities they relate to.

Example 18. *One can distinguish between automation devices (sensors/actuators) and other kinds of equipment. The former would then be defined as Systems and the latter as Platforms, as follows:*

$$
\begin{aligned}
x{:}\mathit{Platform} &\mathrel{:-} x{:}\mathit{WaterTank}.\\
x{:}\mathit{Platform} &\mathrel{:-} x{:}\mathit{PlasticPipe}.\\
x{:}\mathit{Actuator} &\mathrel{:-} x{:}\mathit{Pump}.\\
x[\mathit{actsOnProperty} \Rightarrow \mathit{VolumetricFlowRate}] &\mathrel{:-} x{:}\mathit{Pump}.\\
x{:}\mathit{Actuator} &\mathrel{:-} x{:}\mathit{PneumaticValve}.\\
x[\mathit{actsOnProperty} \Rightarrow \mathit{VolumetricFlowRate}] &\mathrel{:-} x{:}\mathit{PneumaticValve}.\\
x{:}\mathit{Sensor} &\mathrel{:-} x{:}\mathit{FloatSwitch}.\\
x[\mathit{observes} \Rightarrow \mathit{Height}] &\mathrel{:-} x{:}\mathit{FloatSwitch}.\\
x{:}\mathit{Sensor} &\mathrel{:-} x{:}\mathit{UltrasonicSensor}.\\
x[\mathit{observes} \Rightarrow \mathit{Height}] &\mathrel{:-} x{:}\mathit{UltrasonicSensor}.
\end{aligned}
$$

Instances of FloatSwitch and UltrasonicSensor both measure the level of water in a tank, either as a binary value or as a decimal.

Given rules like in Ex. 11 about the deformability of fluids, one can discover potential interactions between e.g. a float switch and a valve, since opening a valve mechanically lowers the level of water in some tank. Like in the previous BA experiment, it is assumed assume that some factory plant or circuit description is available as DL assertions, after transformation from another machine-readable format (as in Fig.

13b). Again, all RDF files are available online[12]. In total, five systems could be identified, formed by combining servient capabilities (and thus, by making them interact). These systems are the following:

VALVE CONTROL An open/close or proportional valve is coupled to a water level sensor to avoid overflow. When water level in a tank goes above a certain threshold, the valve opens.

PUMP CONTROL A water pump is coupled to a water level sensor to refill a tank when necessary. When water level in a tank goes below a certain threshold, the pump starts.

HEATER CONTROL A temperature sensor is coupled to a heater to maintain water at a stable temperature by turning on and off heating (thermostat).

CIRCUIT ANOMALY DETECTION A flow meter and a valve are synchronously monitored to detect potential anomaly in a circuit, e.g. when the measured flow is not null but the valve is closed.

WATER CIRCULATION A pump and a valve are synchronously activated to keep water flowing in a closed loop, e.g. for cleaning purposes.

Each system shall result from the interactions of two servients, each exposing (at least) one 'thing'. These systems can be themselves combined to perform more elaborate (and more realistic) tasks; the formalization of semantic discovery presented here would still cover these cases.

Semantic discovery ran for the same CQ as in the Intel Labs setup (with relatesToProperty). One obtains eight edges in the graph of interactions. All five compound systems can be instantiated, 'Water Circulation' can even be instantiated in two different ways and 'Valve Control' three times. The whole graph of interactions is showed in Fig. 14. In practice, this graph can be used for various purposes, such as the identification of critical points in the network (nodes with a high degree), in a similar fashion to what was described with Intel Labs motes. Here, one of the nodes has a degree of seven, for a maximum degree of 16. If the servient is decommissioned and removed from the network, half of the discovered systems would stop functioning.

3.5 SUMMARY

In this chapter, Web ontologies were formalized as sets of logical rules with classical DL semantics. On this basis, different reasoning tasks can be defined to extend a knowledge graph with implicit knowledge. The reasoning task of interest in WoT is that of *query answering*, which consists in finding possible substitutions for a DL expression with variables that are satisfied by a set of rules. The semantics of WoT

12 https://github.com/vcharpenay/urdf-store-exp

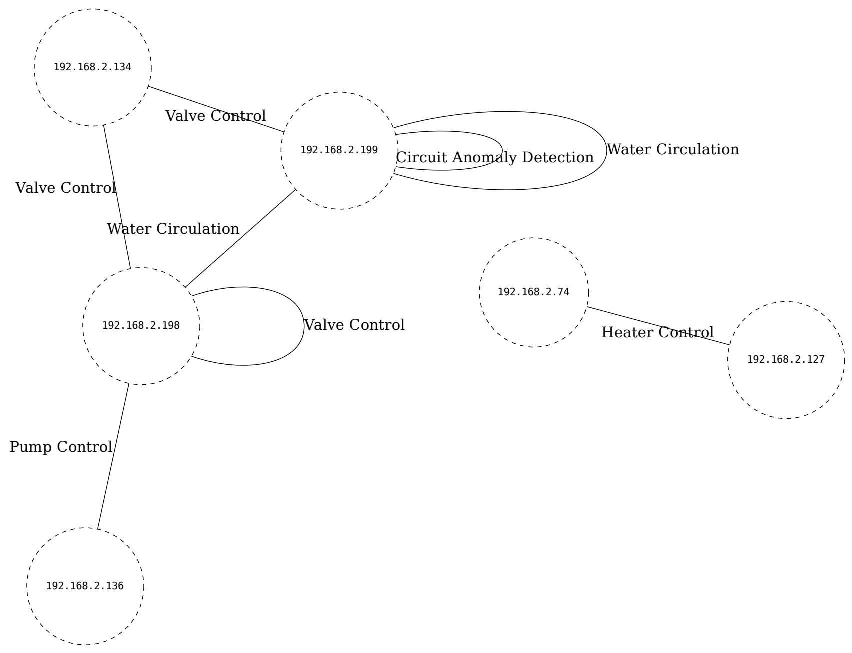


Figure 14: Graph of interactions for the water management plant model of 13; servients are identified by their IP address

interactions for a graph of interactions $G = \langle V, E \rangle$ and a knowledge base $\mathcal{K}$ (a knowledge graph with rules) was defined as a mapping from some edge $e \in E$ to a DL expression f that connects two Things such that $\mathcal{K} \models f$ and f is *minimal* for $\mathcal{K}$ (Def. 11).

From this definition, one can define a discovery task which consists in computing the biggest graph of interactions one can obtain for a query $\mathcal{Q}$ and a knowledge base $\mathcal{K}$. This kind of semantic discovery reduces to query answering. I implemented it for two use cases: first, in a BA system originally designed for research in sensor networks at Intel Labs and second, in the water management plant that was briefly introduced in Ch. 1. In addition, it could be showed how the formalism for WoT semantics given in this thesis helps monitor a system: on Fig. 12, it was showed that a decrease in the number of sensors does not directly affect "physical" coverage of the Intel Labs open space.

One particularity of the DL formalism that was presented is that it includes *existential* restrictions in DL expressions that state the existence of an entity without knowing its identifier. Reasoning with existential restrictions in WoT allows a system to further extend its perception by inferring the existence of physical world entities that it does not directly observe. It is indeed likely that this kind of inference is necessary to capture the complexity of WoT systems from a limited set of 'things' being exposed by servients.

The state-of-the-art in query answering lacks maturity with respect to existential restrictions. What was first observed is that the set of ontologies identified in Ch. 2 (referred to as the WoT ontology cloud) has only few of them and, more generally, it contains few rules to deal with anonymous entities (qualified role inclusions). It was therefore necessary to introduce some rules to do reasoning with physical bodies (solids, fluids) and their properties (mechanical, thermodynamic, geometric). It is also well-known that the main DL query answering engines do not correctly process queries with anonymous entities. This chapter introduced a skolemization algorithm and identified a DL fragment for which the algorithm is tractable (Alg. 1). The two experimental use cases both require a proper skolemization for the correct answer to be found.

The existential reasoning task developed in this chapter is distinct from cause-and-effect reasoning, in which WoT agents plan certain actions based on desired effects. Contrary to the complementary task of existential reasoning, cause-and-effect reasoning requires a feedback loop to ensure that desired effects are consistent with actual effects of some action. In existential reasoning, the assertions that can be inferred depend only on the 'things' in the system that are described in a TD document. In the next chapter, we get interested in collecting TD documents in order to perform reasoning, in particular in constrained environments.

4 EXCHANGING THING DESCRIPTIONS

4.1 INTRODUCTION

WoT and the IoT borrowed many aspects from the field of ubiquitous and pervasive computing, as we saw in introduction (Sec. 1.2.1). One of the objectives of this field of research is to achieve a form of intelligence by demultiplying the number and type of computational agents. More precisely, ubiquitous computing aims at designing *intelligent systems* with high perception capabilities [94, 100]. The kind of intelligence IoT systems feature is not necessarily the same across systems, though. Currently, the most common IoT architecture revolves around Cloud platforms that implement IoT protocols to let masses of low-power devices stream data to the Cloud, in a unidirectional fashion. In this configuration, intelligence primarily comes from the aggregation of data in one place (e.g. via stream processing [74]). It is a kind of "individual" intelligence which is arguably outside the scope of WoT. Indeed, the decentralized nature of the Web should favor "collective" intelligence that materializes as the sum of the contributions of intelligent agents.

From the many features that may define intelligent systems (like autonomy, adaptability or introspection), the most important one in WoT is arguably *self-awareness*. Self-awareness of WoT servients is necessary (but not sufficient) to achieve self-organization and fulfill some automation task as a multi-agent system. It implies that interacting agents must themselves provide affordances to these interactions. In other words, WoT servients should expose and be able to interpret a TD document themselves.

The experiments that were conducted on semantic discovery rely on the somewhat classical assumption that interlinked documents exposed on WoT are first indexed: TD documents were indexed in a TDir in a similar fashion to how early Web portals worked. Unlike the Web of pages, however, WoT systems are backed by a highly heterogeneous infrastructure ranging from Cloud servers to network gateways and low-power edge devices. In the architecture of WoT, every computational agent may get involved in some interaction (at an application level, as opposed to the lower transport and link levels) [64]. A necessary consequence of this "freedom" in terms of infrastructure is that TD documents may be served by any kind of servient, in particular by MCUs and other low-power devices at the edge of a communication network. In the scope of this thesis, it is especially of interest to look at the exchange of TD documents by MCUs for it is a first step towards (collective) intelligence.

In practice, WoT servients should be able to expose a TD document to be indexed by some TDir but also to query the logical assertions it contains, even in a basic fashion. Yet, logic-based intelligent systems are generally hard to design. In Poslad's *Ubiquitous Computing* [94], the problem is summarized as follows:

> Intelligent systems are closely related to ubiquitous computing (and siblings). Design issues of FOL-based intelligent systems are computational complexity, consistency of axiomatization accross agents.

Web ontologies, by definition, guarantee some consistency across servients in terms of axiomatization but their use comes with verbosity in expressing knowledge. In particular, it is not trivial to efficiently compress IRIs. Moreover, the problem of computational complexity cannot be addressed only in terms of theoretical classes of complexity (like P for polynomials). At the scale of MCUs, various factors impacting memory and power consumption must be taken into consideration. For instance, it is important to know the degree of polynomials characterizing tractability.

This chapter introduces the μRDF store, an RDF store designed for MCUs, technically enabled by a binary format for RDF designed for that purpose. Efficient compaction of RDF graphs is achieved by *contextualizing* the assertions it includes. It is a common leverage in the field of embedded software development, as mentioned already in 1991 and later quoted by Poslad:

> [Computation power] is one resource that crucially affects the semantics of its outputs because contexts are more likely to change in dynamic environments and when resource constrained systems are situated in dynamic environments.

After a review of the (Semantic) Web technologies in use in embedded environments (Sec. 4.2), this binary format for RDF is presented in details (Sec. 4.3), followed by an evaluation of the μRDF store in terms of compaction ratio and feasability for query answering (Sec. 4.4).

4.2 RELATED WORK: THE EMBEDDED SEMANTIC WEB

The review that unfolds in this section is a technological review, to the most part. It includes newly standardized technologies that deal with embedded devices, along with several prototypical works applying these technologies. The flagship contribution in that respect is the

WEB	EMBEDDED WEB
HTTP	CoAP [110]
XML	EXI [106]
JSON	EXI4JSON [92], CBOR [13]
HTML	CoRE Link [109]
RDF	HDT [34, 35, 38], RDF/EXI [71]
JSON-LD	*Binary object notation* (Sec. 4.3.1)
LDP	LDP over CoAP [83]
SPARQL	*Frame matching* (Sec. 4.3.2)

Table 9: Main standards of the (Semantic) Web and their equivalence with Embedded Web technologies

work by Guinard and Trifa in 2009, which settles the term of 'Web of Things' [46]. Yet, most contributions that are, conceptually, the closest to this foundational work progressively developed the alternative notion of 'embedded Web', which is arguably more accurate. Embedded Web technologies have in common that they tend to "imitate" widespread Web technologies like HTTP, with the difference that they explicitely target constrained environments. A few attempts to apply them to the Semantic Web and RDF have also been proposed. The review starts with a general introduction to the Embedded Web and then focuses on these works.

4.2.1 *The Embedded Web*

Standardization for the Embedded Web is mostly driven by the Internet Engineering Task Force (IETF) and its Constrained RESTful Environment (CoRE) working group[1], which has been designing standards targeting constrained IP-connected devices. These standards aim at providing similar functionalities to the "classical" Web technologies, that is, RESTful, hypermedia-driven Web interfaces. The CoRE group has standardized the Constrained Application Protocol (CoAP) [110], the Constrained Binary Object Representation (CBOR) [13] and the CoRE Link format [109]. Together with the Efficient XML Interchange (EXI) format [106], these standards form a binary alternative to the otherwise text-based standards usually involved in Web service implementations [70, 108]. Table 9 shows the equivalence with HTTP, JSON, XML and (to some extent) HTML.

In the present context, a constrained device is a low-power MCU with 8 to 64 kB RAM [12]. Typically, such a device is too constrained

1 https://datatracker.ietf.org/wg/core/charter/

to support standard Web technologies but powerful enough to integrate an IP connectivity stack. For instance, for the most constrained of these devices, IP version 6 can still be achieved given a few optimizations on message sizes. Some of these optimizations have been standardized by the IETF under the name 6LoWPAN[2]. This class of devices gained particular attention in the past decade, especially through the development of dedicated operating systems like Contiki [30] and TinyOS [80]. All recent standards focusing on MCUs allow for novel machine-to-machine interaction patterns involving many devices in highly decentralized systems. In other words, the Embedded Web realizes the MAS vision of WoT that was adopted in this thesis. It is also worth noting that the IRTF (the daughter organization of the IETF dedicated to research) works jointly with the W3C in this direction[3].

The European project SPITFIRE [93] paved the way to the use of RDF and other Semantic Web technologies on the Embedded Web, combining it with architectural principles of WoT. Since then, several methods were proposed to serialize and process RDF data on constrained devices.

4.2.2 *Serializing RDF on the Embedded Web*

Until today, a larg part of research towards storing and querying RDF has focused on very large, static datasets stored on powerful machines, sometimes involving parallel computation. In contrast, storage mechanisms for resource-constrained devices remain mostly unexplored. Until recently, no realistic use case could be found where computational devices had limited resources but still IP connectivity.

The first work that addressed constrained devices is part of SPITFIRE and is called the Wiselib TupleStore [53]. Built on top of Wiselib, a substitute to the C++ standard library designed for embedded systems, the Wiselib TupleStore internally stores IRIs in a tree-shaped data structure to compact them. In SPITFIRE, the Wiselib TupleStore was ported on wireless motes, similar to those used for the Intel Labs sensor network, and RDF graphs were serialized in a binary format called SHDT [52], a streaming (and simplified) version of the Header-Dictionary-Triples (HDT) format.

The original objective of HDT was to compact large RDF datasets e.g. to fit in the main memory of a personal computer. But as a binary format, its compression scheme could reasonably be used on small datasets as well. An HDT document is divided into three sections containing respectively metadata (*Header*), resource IRIs (*Dictionary*) and the triples themselves, indexed by subject (*Triples*). Although most RDF stores also implement a similar partitioning, HDT

2 https://datatracker.ietf.org/wg/6lowpan
3 https://irtf.org/t2trg

is not designed for complex query processing (which usually involves multi-indexing) but rather for unidirectional serialization [35]. HDT datasets are rarely deserialized. One can also add the comment that the header section is optional, thus of little interest in the present context.

In the original proposal for HDT, all triples are merged in an single array while separations between them are stored in a bitmap, easily compressible. HDT achieves high compression ratios compared to classical compression schemes like gzip. An alternative triple indexing method was also proposed, performing vertical partitioning with k^2-tree compression (k^2-triples) [38]. k^2-triples achieves better compression ratios while efficiently processing predicate-bound triple pattern matching queries.

Example 19. *Let us illustrate the (original) HDT and the k^2-triples serializations on the room example of last chapter with wall orientation (Ex. 6). Let us assume the following canonical form for this example (without blank node):*

```
legoland:Site[
  hasSpace → 31.638:Space[
    containsElement → southWall:Wall[
      hasOrientation → south,
      hasSubElement → radiator1:Radiator
    ] and eastWall:Wall[
      hasOrientation → east,
      hasSubElement → radiator2:Radiator
    ]
  ]
].
```

The corresponding dictionary is as follows (in lexixographic order):

1	31.638	(individual names occuring at least twice)
2	eastWall	
3	radiator1	
4	radiator2	
5	southWall	
6	legoland	(other individual names)
6	east	(class names and individual names occurring only as value)
7	south	
8	Radiator	
9	Site	
10	Space	
11	Wall	
1	`rdf:type`	(property names)
2	containsElement	
3	hasOrientation	
4	hasSpace	
5	hasSubElement	

The triples section for HDT is composed of two integer streams for properties (including `rdf:type`) and values:

Properties	1 2 0 1 3 5 0 1 0 1 0 1 3 5 0 1 4
Values	10 0 2 5 0 11 0 6 0 4 0 8 0 8 0 11 0 7 0 3 0 9 0 1

These streams must be read as follows: start by a dictionary look-up and take the first available individual (1 – 31.638); then read the first integer in the properties stream (1 – `rdf:type`) and read every integer in the values stream until 0 is found (10 – Space); read the next integer in the properties stream (2 – containsElement) and read the values stream until 0 is found (2 5 – eastWall, southWall); repeat until 0 is found in the properties stream; take the next individual in the dictionary (2 – eastWall); repeat until both streams are empty.

The k^2-triples serialization indexes triples by properties instead of individuals. For each property appearing in the RDF graph, a bitmap like the following is constructed (here, for containsElement):

0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Rows correspond to individuals and columns are values for the containsElement *property. Matrix indices are given by the same dictionary as for HDT, with "padding" to the closest power of two. Here,* 31.638 *(first row) contains the elements* eastWall *(second column) and* southWall *(fifth column). Compression is achieved by spatially dividing the bitmap in four squares and reducing uniform parts to a single bit, as follows:*

1000 1100 1000 1000 0100 1000

The first four bits (1000) encode whether each 8×8 square of the bitmap is uniform (0) or not (1) in the following order: top left, top right, bottom left, bottom right. For each non-uniform square, read the next four bits (1100) encoding the same information for sub-parts of size 4×4; repeat with sub-parts of size 2×2. This data structure, called k^2-trees is, often used to compress geo-spatial data. In the present example, only two squares of size 2×2 are not uniform. The rest can be serialized in a couple of bits.

HDT and k^2-triples show excellent results in terms of compression and query processing speed (for simple queries). However, a recent study suggests that there exists a better alternative for small datasets and suitable for embedded devices [71]. The study presents datasets of semantically annotated sensor measurements where an EXI serialization of RDF/XML data is more compact than HDT. The two approaches (EXI and HDT) have been combined in a proposal called Efficient RDF Interchange (ERI) [34], which, however, primarily addresses data streams with time considerations, which are not of relevance here.

More recently, an RDF store prototype following similar principles was introduced, with special care given to loading secondary storage data into RAM for efficient query processing with a limited amount of RAM [75]. However, this prototype called RDF for Lightweight Edge Devices (RDF4Led), still targets higher classes of devices than what is considered in this thesis, with typically 512 MB to 1 GB RAM.

Finally, an interesting work worth mentioning is the Linked Data Platform (LDP) specification [113] and its mapping to CoAP [83]. LDP is a Web architecture to expose RDF graphs in a resource-centric way, where assertions about individuals can be separately retrieved via a Web interface by dereferencing each individual's IRI. Constrained node networks are seen as one of the application fields of LDP (over CoAP) for this architecture offers a means to exchange low amounts of data at once regardless of the serialization format [7]. However, we will see in Sec. 4.4 that LDP on constrained nodes proved impractical due to higher costs for data exchange than for local query processing.

Table 9, in addition to what we have just discussed, includes Embedded Web equivalences for JSON-LD and SPARQL. These contributions to the state-of-the-art are my own, although they are based on the technologies that were presented in this review. Both contributions rely on a formalization of the JSON-LD data model at the basis of the µRDF store. This formalism is presented next.

4.3 THE µRDF STORE: RDF ON MICRO-CONTROLLERS

The µRDF store is an embedded RDF store designed to store assertions in the main memory of an MCU, as well as to query and update these assertions. It is based on a binary object notation for RDF sufficiently compact to fit in a small segment of RAM. This notation also inspired the human-readable notation used in the examples of this thesis.

We can already observe both in the DL notation of the thesis and the HDT serialization format that triples are "factorized" by individual and then by property. This practice is very common and it is at the basis of Turtle, the Terse RDF Triple Language. It is also the basis for an object representation of triples which greatly facilitates the integration of the RDF data model with object-oriented programming languages. This representation was recently standardized as JSON-LD [114]. Both Turtle and JSON-LD allow for prefix declarations, which greatly reduce the size of a serialization, but JSON-LD even allows full IRIs to map to arbitrary character strings. It then becomes possible to *compact* IRIs in an RDF graph by defining a proper mapping to all the IRIs it contains. Such a mapping is called a JSON-LD *context*. A formalism for context-based compaction is presented in the following.

This formalism and the compaction experiments that were conducted alongside were first published in June 2018 [23]. To the best of my knowledge, it was the first attempt to formalize (parts of) JSON-LD. This work was significantly extended since then.

4.3.1 *JSON-LD Expressions and Semantics*

The DL notation of the thesis is borrowed from F-logic, an extension of FOL for object-oriented programming [65]. FOL defines terms and formulas. F-logic extends the definition of formulas to include class membership (:) and property/value pairs ([. . .]). N^C, N^P, N^I respectively denote the mutually disjoint sets of class names, property names and individual names and V, the set of variables (equivalent to blank node identifiers). U then refers to the set of UTF-8 strings. Necessarily, we have $(N^C \cup N^P \cup N^I) \cup V \subset U$, since every IRI is also a character string, and $(N^C \cup N^P \cup N^I) \cap V = \emptyset$.

Definition 12. *Let* $t \in U$ *and* $t', t_1, \dots, t_n \in U \setminus V$. *A JSON-LD expression is a formula of the form*

$$t{:}t'[t_1 \to f_1,\ \dots,\ t_n \to f_n].$$

such that $f_1, \dots, f_n$ *are conjunctions of JSON-LD expressions. A JSON-LD graph is a set of JSON-LD expressions.*

Several examples of JSON-LD formulas have already been given in this thesis. If we exclude existential restrictions, Def. 3 is a special case of the present definition, where $t, f_1, \dots, f_n$ are restricted to individual names and t' to a single class name. Moreover, in Exs. 5 & 6, variables (or blank nodes) were omitted since they appeared only once in the formula: their position in the formula is enough to identify them.

We denote F the set of all formulas defined by Def. 12. One can already observe that not all $f \in F$ has a corresponding RDF representation, since terms may not be IRIs. This can be alleviated if we introduce the notion of JSON-LD context.

Definition 13. *A JSON-LD context is a function* $c : U \mapsto N^C \cup N^P \cup N^I \cup V$ *such that for all term* $t \in N^C \cup N^P \cup N^I \cup V$, *we have* $c(t) = t$ *and* c *is an injection (for all* $t, t' \in U$ *distinct, we have* $c(t) \neq c(t')$*).*

The EXPAND *procedure defined for a JSON-LD expression* f *and a context* c *is the application of* c *on all terms of* f. *The* COMPACT *procedure is the reverse transformation, i.e. such that* $\text{COMPACT}(\text{EXPAND}(f, c), c) = f$.

In Def. 13, restrictions apply to c for the sake of simplicity, so that EXPAND and COMPACT are exactly the inverse of each other. However, these two procedures as defined by the JSON-LD W3C standard do not have this restriction [82]. In the present definition, RDF literals are also ignored as they are of lesser relevance here.

Example 20. *Every example given so far (including in Ch. 3) is in fact a* compacted *form of some expression only with IRIs. For instance, Ex. 19 is the compacted form of the following expression:*

$\texttt{<tag:legoland>}{:}\texttt{bot:Site}[$

$\quad \texttt{bot:hasSpace} \rightarrow \texttt{<tag:31.638>}{:}\texttt{bot:Space}[$

$\quad\quad \texttt{bot:containsElement} \rightarrow \texttt{<tag:southWall>}{:}\texttt{ex:Wall}[$

$\quad\quad\quad \texttt{ex:hasOrientation} \rightarrow \texttt{<tag:south>},$

$\quad\quad\quad \texttt{bot:hasSubElement} \rightarrow \texttt{<tag:radiator1>}{:}\texttt{ex:Radiator}$

$\quad\quad]$ and $\texttt{<tag:eastWall>}{:}\texttt{ex:Wall}[$

$\quad\quad\quad \texttt{ex:hasOrientation} \rightarrow \texttt{<tag:east>},$

$\quad\quad\quad \texttt{bot:hasSubElement} \rightarrow \texttt{<tag:radiator2>}{:}\texttt{ex:Radiator}$

$\quad\quad]$

$\quad]$

$].$

Recall that `tag:` is a URI scheme (Table 6) while `bot:` and `ex:` are namespace prefixes, i.e. shorthands for full IRIs (Table 13).

In fact, one single context c can be defined for the whole thesis, such that every class, property or individual name defined in an ontology under a certain namespace is represented in a compacted way by its local name (without namespace). A definition of c is given in Appendix B. For a finite set of ontologies, processing a JSON-LD expression that includes only IRIs or its compacted form is semantically equivalent, as per the following definition:

Definition 14. *Let f be a JSON-LD expression. Let c be a JSON-LD context and f' the JSON-LD expression such that $f' = \text{EXPAND}(f, c)$. We say that an interpretation $\mathcal{I}$* contextually *satisfies f and write $\mathcal{I} \models_c f$ if:*

- *f' is a DL expression as per Def. 3*
- *$\mathcal{I} \models f'$ as per Def. 4*

Let $\mathcal{G}$ be a JSON-LD graph. We also have $\mathcal{I} \models_c \mathcal{G}$ if for all f in $\mathcal{G}$, $\mathcal{I} \models_c f$.

Definition 14 allows one to abuse notation throughout this thesis. Indeed, it immediately follows that for all DL expression f, if $\mathcal{I} \models_c \text{COMPACT}(f, c)$, then we have $\mathcal{I} \models f$ and both notations (compacted and expanded) can be used interchangeably for a fixed context. The JSON-LD semantics are consistent with the well-defined RDF semantics [48, 54]: JSON-LD and RDF graphs that are syntactically equivalent are also equivalent semantically.

Now, we can also think of a context that *minimizes* the size of all strings in an expression, with no care for human-readability. That is, it is always possible for a fixed set of IRIs N to construct a context $c_{\min}$ that maps the shortest $|N|$ UTF-8 strings to elements of N.

Example 21. *Let us take Ex. 20 again.We have:*

$$N = \{\texttt{<tag:legoland>,bot:Site,bot:hasSpace,<tag:31.638>,...}\}$$

from which c_{min} *can be constructed. The output of compaction with* c_{min} *is the following:*

$$a{:}b[c \rightarrow d{:}e[f \rightarrow g{:}h[i \rightarrow j,\ k \rightarrow l{:}m] \text{ and } n{:}h[i \rightarrow o,\ k \rightarrow p{:}m]]].$$

One can already notice that the above example is close to the HDT format. Besides discrepancies in indexing IRIs, the main difference is that HDT does not have any form of "nesting" formulas in one another. To get even closer to HDT, *flattening*, the last transformation defined by the JSON-LD processor specification, is now introduced [82].

Definition 15. *The* FLATTEN *procedure defined for a JSON-LD expression* f *is the construction of a JSON-LD graph* $\mathcal{G} = \text{FLATTEN}(f)$ *such that:*

- *for all* $\mathcal{I}$, c, *we have* $\mathcal{I} \models_c f$ *if and only if* $\mathcal{I} \models_c \mathcal{G}$
- *for all* $f' \in \mathcal{G}$, *of the form* $t'{:}t''[t'_1 \rightarrow f'_1, \ldots t'_n \rightarrow f'_n]$ *as per Def. 12,* f'_i *is a conjunction of terms for* $i \in [1, n]$ *(no nesting)*

Example 22. *The result of applying* FLATTEN *to Ex. 21 is as follows:*

$$\begin{aligned}
&a{:}b[c \rightarrow d].\\
&d{:}e[f \rightarrow g \text{ and } n].\\
&g{:}h[i \rightarrow j,\ k \rightarrow l].\\
&l{:}m.\\
&n{:}h[i \rightarrow o,\ k \rightarrow p].\\
&p{:}m.
\end{aligned}$$

The last step towards a binary representation of JSON-LD expressions is to turn string-based tokens into numeric tokens to then be efficiently serialized in a binary format, like those mentioned in Sec. 4.2 (EXI4JSON, CBOR).

Example 23. *If we replace the string-based tokens {:, [, →, ,,], and} with numeric tokens (e.g. 1, 2, 3, 4, 5, 6) in the flattened expression of Ex. 22, we obtain:*

a 1 b 2 c 3 d 5 6
d 1 e 2 f 3 g 6 n 5 6
g 1 h 2 i 3 j 4 k 3 l 5 6
l 1 m 6
n 1 h 2 i 3 o 4 k 3 p 5 6
p 1 m

One can note that the HDT representation of Ex. 19 is shorter than that of Ex. 23 (42/53 symbols). However, HDT must also include the dictionary (at least 11 symbols) to correctly deserialize the properties and values streams. In contrast, a context c_{min} can be defined globally so that it is shared by all servients involved in a WoT system, which relieves them from sending the context every time JSON-LD expressions are exchanged. In practice though, c_{min} cannot be as optimal as the context defined in Ex. 21, which is specific to a single formula. Instead, it can be constructed from the vocabulary contained in a fixed CBox $\mathcal{C}$, that is $N_{\mathcal{C}}^{C} \cup N_{\mathcal{C}}^{P} \cup N_{\mathcal{C}}^{I}$. Finally, one can also note that a binary serialization of nested expressions (like Ex. 21) would be even smaller in size, as opposed to a flattened form. However, I empirically observed that the impact on compaction was not significant while a flattened form greatly facilitates the processing of serialized JSON-LD.

Later in this chapter, experimental results on the performances of this compaction technique based on a minimal context will allow one to have a more formal discussion on these aspects (Sec. 4.4). Before that, we can move to the details of the µRDF store's query answering capabilities, which are based on JSON-LD *framing*. The underlying principle is that queries (JSON-LD frames) are themselves JSON-LD expressions and can therefore be stored efficiently as well, so that there is enough RAM left to store intermediate results.

4.3.2 *JSON-LD Frame Matching*

JSON-LD framing is not part of the official W3C recommendation. The formalism presented in the following is based on the latest community draft for JSON-LD 1.1, as of December 2017 [115]. JSON-LD framing includes two aspects: frame matching and re-shaping, analogous to SPARQL's SELECT and CONSTRUCT query types. In the following, we will only consider frame matching, which is very similar to DL query entailment.

Definition 16. *Let $\mathcal{G}$ be a JSON-LD graph and $\mathcal{F}$ another JSON-LD graph referred to as a frame. Let c be a JSON-LD context and $\mathcal{G}', \mathcal{F}'$ JSON-LD graphs such that $f \in \mathcal{G}$ if and only if $f' = \text{EXPAND}(f, c) \in \mathcal{G}'$, as well as $f \in \mathcal{F}$ if and only if $f' = \text{EXPAND}(f, c) \in \mathcal{F}'$. We say that $\mathcal{G}$ contextually entails (or matches) $\mathcal{F}$ and write $\mathcal{G} \models_c \mathcal{F}$ if $\mathcal{G}' \models \mathcal{F}'$ as per Def. 5.*

In the W3C specification, framing also includes disjunctions, via a 'require all' flag, while the present definition only includes conjunctions of expressions. Every query involving both constructs can always be turned into a normal form, i.e. a disjunction of conjunctions. In my original formalization of JSON-LD, I showed that this normalization can be performed in polynomial time [23]. We can therefore leave this aspect aside.

Moreover, in the W3C specification, frames can also include special variables, 'none' and 'wildcard', which are meant to match respectively none of the JSON-LD expressions and all of them. However, introducing these variables in the present formalism presents two issues. First, it would require including negation in formulas, which was explicitly left aside in the previous chapter. Second, correct semantics for JSON-LD framing would come with the assumption that any assertion that is not in $\mathcal{G}$ is false. However, DL semantics usually has a more permissive behavior towards absent assertions: it only considers them as *undefined*. They are respectively known as the Closed and Open World Assumptions. To avoid mixing both in the same formalism, 'none' and 'wildcard' were left aside.

To conclude with theoretical considerations, we would need an insight into the complexity of JSON-LD frame matching. Since JSON-LD and RDF are equivalent representations, the entailment problem of Def. 16 has the same complexity as RDF conjunctive query entailment: it can be solved in polynomial time for the data complexity [48]. It then follows that query answering, the problem of finding all substitutions for a frame (or its equivalent CQ), can also be solved in polynomial time, as it can be reduced to query entailment after substituting variables in a CQ to all individuals in the graph.

The MATCH procedure (Alg. 2) computes all answers for some input graph $\mathcal{G}$ and a frame $\mathcal{F}$. Both inputs must be flattened graphs, i.e. the result of applying FLATTEN to all of the expressions they include. The answers returned by MATCH are *mappings*, which are partial functions defined in a set-theoretical fashion.

Definition 17. *[48] A mapping μ is a partial function whose domain, denoted dom(μ), is a subset of V (the set of variables) and whose codomain is U (the set of UTF-8 strings, including variables).*

Let μ_1, μ_2 be mappings. If for all $x \in dom(\mu_1) \cap dom(\mu_2)$, we have $\mu_1(x) = \mu_2(x)$, then the union of μ_1, μ_2, denoted $\mu_1 \cup \mu_2$, is the mapping μ such that $dom(\mu) = dom(\mu_1) \cup dom(\mu_2)$.

Let $\mu_\emptyset$ be the mapping such that $dom(\mu_\emptyset) = \emptyset$. For all μ, $\mu_\emptyset \cup \mu = \mu$

Mappings were first introduced to define an algebra for SPARQL [48] but it is also a convenient tool to define frame matching. Constructing mapping unions is analogous to performing *joins* in relational databases. Here, matching consists in joining solution sets obtained by applying MATCHEXPRESSION on $\mathcal{G}$ for each (flattened) expression in $\mathcal{F}$. The MAP procedure, defined for two JSON-LD expressions f and f', returns a set of mapping Ω such that for all $\mu \in \Omega$, $\text{dom}(\mu)$ is the set of variables in f and the expression obtained by replacing all x in f with $\mu(x)$ is included in f'. It is easy to see that if $\text{MAP}(f, f') \neq \emptyset$, then $\{f'\} \models_c \{f\}$ for all c. This property guarantees the correctness of Alg. 2.

As one can note, JSON-LD framing can be computed regardless of the context. In particular, matching can done on frames and graphs

Algorithm 2 Frame matching algorithm for a flattened graph $\mathcal{G}$ and a flattened frame $\mathcal{F}$.

```
function MATCH($\mathcal{G}$, $\mathcal{F}$)                ▷ join mappings between expressions
    if $\mathcal{F} = \emptyset$ then
        return $\{\mu_\emptyset\}$
    else
        Let $f \in \mathcal{F}$ be some expression
        Let $\Omega = \text{MATCH}(\mathcal{G}, \mathcal{F} \setminus \{f\})$
        Let $\Omega' = \text{MATCHEXPRESSION}(\mathcal{G}, f)$
        for all $\mu \in \Omega$ do
            $\Omega := \Omega \setminus \{\mu\}$
            for all $\mu' \in \Omega'$ do
                if $\mu'' = \mu \cup \mu'$ is a mapping then
                    $\Omega := \Omega \cup \{\mu''\}$
                end if
            end for
        end for
        return $\Omega$
    end if
end function
function MATCHEXPRESSION($\mathcal{G}$, f)              ▷ match single expression
    if $\mathcal{G} = \emptyset$ then
        return $\emptyset$
    else
        Let $f' \in \mathcal{G}$ be some expression
        Let $\Omega = \text{MAP}(f, f')$
        return $\Omega \cup \text{MATCHEXPRESSION}(\mathcal{G} \setminus \{f'\}, f)$
    end if
end function
```

compacted with c_{min}, such that MCUs can exchange and store compacted forms only. I implemented the µRDF store on this basis. In case the input frame $\mathcal{F}$ and the input graph $\mathcal{G}$ are compacted with different contexts c and c′, frame matching may return false positive. However, it is always possible to compute a frame $\mathcal{F}'$ such that $f \in \mathcal{F}$ if and only if $f' = \textsc{compact}(\textsc{expand}(f, c), c') \in \mathcal{F}'$. This operation may be too costly for MCUs if c or c′ are large. It can be assumed that this "context switching" operation is done on an unconstrained machine that acts as intermediary in cross-domain applications.

4.4 EVALUATION

The main objective behind this formalization of JSON-LD is to provide a representation for RDF that could be stored and processed by MCUs. Although this approach does not make particular assumptions about the nature of assertions to be stored and processed, the evaluation exposed next concentrates on assertions likely to be found in TD document to drive WoT-related applications. All datasets and context files can be found online[4]. In the following, the expression 'µRDF store' refers to the computer program to manage compacted JSON-LD data.

The performances of JSON-LD compaction were first evaluated in comparison to the state-of-the-art on WoT-related datasets and then applied to semantic discovery, on the water management use case of last chapter. Results of this evaluation are provided next (Secs. 4.4.1 & 4.4.2). Finally, a benchmark that was designed for JSON-LD frame matching and tested on the µRDF store is introduced (Sec. 4.4.3).

4.4.1 *Binary JSON-LD & Compaction*

4.4.1.1 *Approaches & Dataset Description*

As underlined in Sec. 4.2, the state-of-the-art in binary formats for RDF includes two main approaches: HDT (in its two variants) and RDF/EXI. JSON-LD compaction allows for a third alternative: encoding JSON-LD in its compacted form (using a global context) in a binary JSON format. The binary formats that were selected for these experiments are EXI for JSON (EXI4JSON) and CBOR. EXI4JSON is an extension of EXI to represent JSON documents in binary XML [92].

As mentioned in Sec. 4.3.1, building a minimal context is somewhat arbitrary. For instance, one could choose either to include all ontological concepts exposed on the Semantic Web in a single context or tailor an application-specific context covering a limited set of ontologies (a CBox). In the experiments, two contexts were generated for every dataset: the "optimal" context c_{min} constructed from the

4 https://github.com/vcharpenay/urdf-store-exp/

	BTCSAMPLE	NODE	SSP	DESIGO
$\|\mathcal{A}\|$	174	73	4859	84908
$\|N^I_{\mathcal{A}}\|$	174	26	1345	21527
n, s.t. $\mathcal{A} = \bigcup_i^n \mathcal{A}_i$	-	-	313	1843
$\text{avg}_i(\|\mathcal{A}_i\|)$	-	-	15	70
$\text{min}_i(\|\mathcal{A}_i\|)$	-	-	13	5
$\text{max}_i(\|\mathcal{A}_i\|)$	-	-	19	662

Table 10: Statistics on datasets designed for compaction experiments; $|\mathcal{A}|$ denotes the number of type and property statements (i.e., triples) in ABox $\mathcal{A}$ while $|N^I_{\mathcal{A}}|$ is the number of distinct entity names in $\mathcal{A}$

IRIs included in the dataset only (for comparison purposes) and a context generated from all IRIs included in the Web ontologies referenced by the dataset. Compaction using the latter context will not perform as good as with the optimal context but it better estimates the performances one should expect in a production environment.

As a result, seven approaches were compared: (1) HDT, (2) k^2-triples, (3) RDF/EXI, (4) EXI4JSON, (5) CBOR, (6) EXI4JSON with an optimal context and (7) CBOR with an optimal context.

The compaction performances of the seven approaches were compared on four data sets: a sample from the Billion Triples Challenge (BTCSAMPLE), a single sensor measurement (NODE), the output of a proxy service for sensor data (SSP) and an export of a building model from the Siemens Desigo CC platform (DESIGO). The first three datasets were provided by Hasemann et al. in the context of SPITFIRE [53]. The last one was added to provide a comparison on real-world data, exported from a production environment. Statistics on these datasets are given in Table 10.

The BTCSAMPLE dataset includes 174 random assertions from a large social network. Each individual object has a single property. IRIs reference a large variety of ontologies, such as FOAF, GeoNames[118] or the Simple Knowledge Organization System (SKOS) [86]. The main purpose of BTCSAMPLE as a test set is to evaluate how the different approaches perform on IRI compaction, regardless of the data structure.

In contrast to BTCSAMPLE, NODE and SSP mostly use SSNX to express the semantics of sensor measurements, along with QUDT and SWEET for quantity kinds and units. NODE is a small dataset produced by one single sensor while SSP includes the measurements of hundreds of sensing devices. The latter shows many redundances in the data. Both NODE and SSP are realistic WoT datasets.

The last dataset, which is denoted DESIGO, was generated from a simulated building managed by the Siemens Desigo CC platform, at a real scale. The data is exported from Desigo CC as a collection of TD documents that encapsulate properties like room temperature and sensor readings, actions like start/stop command on ventilation or events like fire alarms. The ontologies referenced by these TD documents are an OWL export of the Desigo data model, which is an object-oriented model close to BACnet, and SAREF.

The dataset, which includes around 10,000 data points, is the sum of 1843 interlinked TD documents. Because TD documents are meant to be individually exposed by WoT servients, DESIGO was considered both as a whole and as an aggregation of separate documents, individually compacted and serialized. Similarly, SSP is an aggregation of individual sensor measurements, stored independently on hundreds of sensors. After looking at compaction on whole datasets, the "piecewise" distribution of datasets were considered, where each piece is theoretically managed by a distinct servient.

4.4.1.2 *Results*

Results are shown on Fig. 15. What the results first show is that k^2-triples, although highly efficient on large datasets, performs poorly on small ones: on NODE, the size for k^2-triples is 16kB while all others remain under 3kB. Moreover, on all datasets, RDF/XML is outperformed by other approaches. In particular, it is outperformed by HDT, contrary to what earlier results suggested [71]. The difference in performances most likely rests on the fact that EXI performs good on datasets with many non-string literals (e.g. numeric sensor values), which happens not to be the case in these datasets. It is interesting to note, however, that EXI performs better than CBOR on compressing redundant structures (as in SSP).

HDT was originally designed to compress very large datasets but it also performs good on medium size datasets, like SSP and DESIGO. It is more than twice as performant as EXI4JSON and by far better than CBOR (which is consistently outperformed by EXI4JSON). However, on the small NODE dataset, EXI4JSON and HDT have comparable results, regardless of the context used. The results on BTCSAMPLE illustrate the impact of choosing a context on the overall compaction performances: the optimal context allows for 37% compaction compared to the ontology-based context (5409 kB / 8639 kB). Indeed, the higher the variety of IRIs used in an application, the more arduous it is to design a context that fits.

JSON-LD compaction shows best results on datasets of less than hundred triples, the typical size of a TD document stored by a constrained WoT agent. Figure 16 show how EXI4JSON and CBOR are more efficient than HDT on SSP and DESIGO, when pieces of data are serialized separately (using the same ontology-based context). For median results on DESIGO, EXI4JSON and CBOR achieve a compaction ratio of 58% and 50% compared to HDT (respectively). Com-

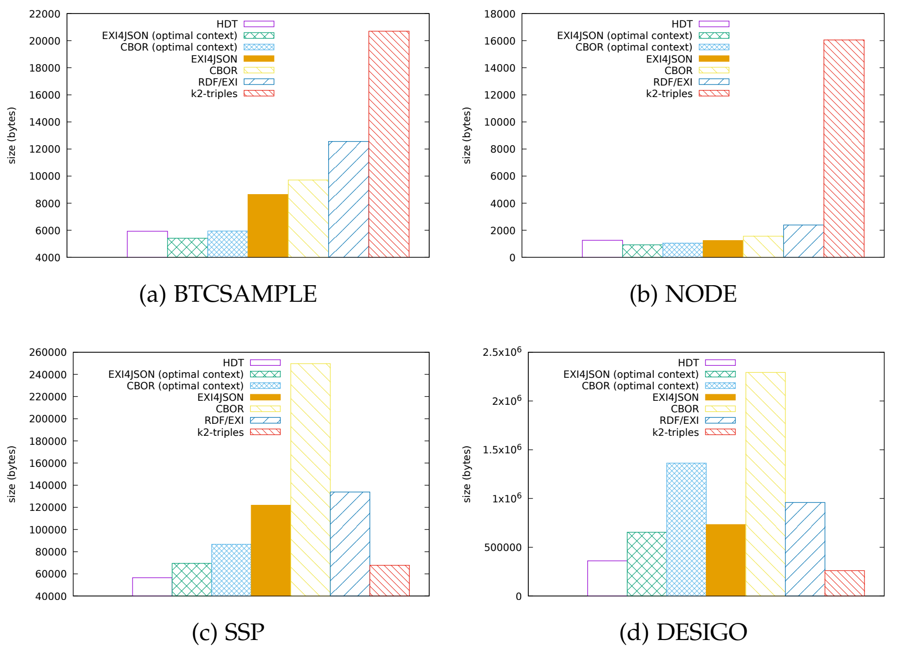


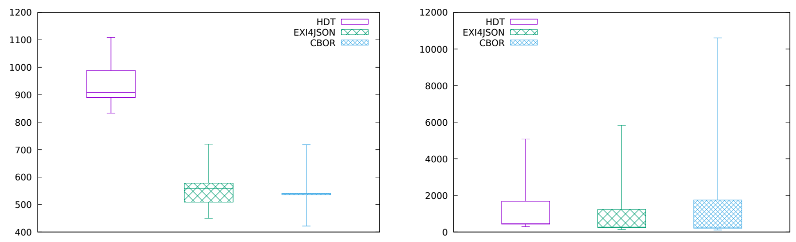
Figure 15: Size of datasets serialized in compact binary formats; datasets are considered as a whole

paction ratios on SSP are similar: 62% and 59%. Interestingly, CBOR performs better on median results but it shows a higher variance than EXI4JSON. One can also note that the DESIGO dataset appears skewed towards small TD documents. The reason is that many TDs in the dataset are logical entities, with no explicit affordance attached to them (and thus with only few assertions). It is the case for e.g. buildings, floors and rooms.

The overhead of HDT on small datasets is due to the dictionary it embeds in every individual document (mostly redundant). Strictly speaking, the principle of defining a global context for a set of documents could also apply to HDT dictionaries. However, splitting a dictionary into global and local parts would lower the compaction ratio and require decompression on the MCU before processing (e.g. to match a JSON-LD frame).

4.4.2 *Semantic Discovery Exchanges*

Observing that EXI4JSON consistently outperforms other serialization formats for small datasets, it was chosen for the semantic discovery experiments of last chapter (Sec. 3.4). In both use cases (Intel Labs and water management), MCU servients exchange ABox assertions with a TDir instance. A possible indicator to look at are payload sizes in these exchanges when assertions are serialized in EXI4JSON over CoAP. The first exchange required for discovery is the registra-



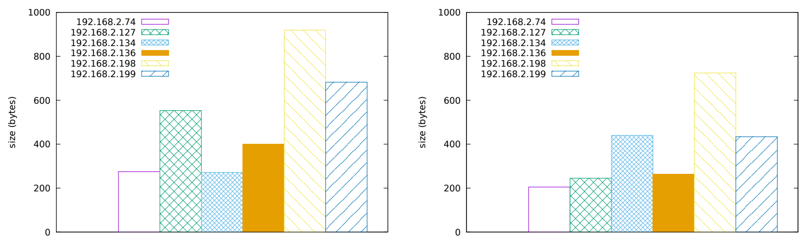
(a) SSP (b) DESIGO

Figure 16: Distribution of size values for sets of individual documents serialized in compact binary formats; datasets are considered as an aggregation of individual documents serialized separately

tion of all TD documents stored by servients on the TDir. Registration can either happen by `POST` requests sent by individual servients to the TDir or by a `GET` request broadcast by the TDir to all µRDF store instances in the network. After query answering terminates, the TDir must send update ABoxes back to the servients, as per Definition 10. Updates are performed as `PUT` requests sent by the TDir to each µRDF store.

In the Intel Labs use case, the only difference across mote descriptions is their location, which accounts only for a few bytes when serialized. Provided a minimal context with schema.org and BOT terms, the EXI4JSON serialization of a mote TD document is of 80 bytes only (in average). However, because of large fully connected components in the graph of interactions, the udate ABox a mote receives is of (maximum) 419 bytes. It is interesting to note that if every mote involved in the biggest connected component (16 nodes) only receives 419 bytes, the TDir sends 16 times this amount of data, which is entirely redundant. However, after discovery, each mote is autonomous insofar as it can choose with which other mote to communicate. This self-organization property is crucial in sensor network applications.

Fig. 17 shows payload sizes for the water management use case. In the case where a servient provides affordances to more than one 'thing' (as for IPs 192.168.2.198 and 192.168.2.199), TD documents are merged and sent in a single payload. The JSON-LD context used here was constructed from eCl@ssOWL, schema.org, OM, SOSA and SSN. By comparing the two histograms, we can see that the size of updates is comparable to the size of the original TD documents. Since servients discover interactions with one or two other servients only (Fig. 14), they turn out to roughly exchange each other's TD documents. All payload sizes fit in a single CoAP block (of maximum size of 1,024 bytes), which represents no technical challenge for MCUs like the ESP8266.



(a) TD documents (b) Update ABoxes

Figure 17: Size of ABox assertions exchanged between servients and Thing Directory, serialized in EXI4JSON; servients are identified by their IP address

4.4.3 *JSON-LD Framing Benchmark*

After discovery terminates, MCU servients should be able to query their own knowledge base to select potential interaction partners, e.g. with the query $x[\mathsf{relatesToProperty} \to z]$ and $y[\mathsf{relatesToProperty} \to z]$. It is unrealistic to consider full reasoning support on an MCU, as underlined in introduction to this chapter. However, the ESP8266 could perform simple pattern matching without inference, provided that assertions are already "materialized" in its local knowledge base. With this in mind, a procedure for JSON-LD frame matching can be formalized, as shown in Alg. 2. This aspect is evaluated next on a benchmark.

There exists several benchmarks for the related problem of SPARQL query answering. Among them, the most cited are the Lehigh University Benchmark (LUBM) [47] and the Berlin SPARQL Benchmark (BSBM) [10]. The former focuses on query answering with OWL while the latter was tailored to evaluate scalability on a very large dataset. Both provide synthetic dataset generators that can be adapted to the needs of the thesis, to generate (small) datasets comparable to those used in the semantic discovery experiments. Because most BSBM queries are of lesser interest on datasets of less than a million assertions, it was chosen to evaluate frame matching against LUBM, on a dataset of 529 triples.

LUBM is composed of a query mix of 14 queries of various complexity to retrieve information about universities, students, faculty members and lectures. Queries also have different selectivities. For instance, Q2 is designed not to return any answer while Q5 has many answers (low selectivity). All queries but two (Q6, Q14) involve joining intermediate results, what was called *complex* queries earlier in the thesis. A documentation of the whole benchmark can be found online[5]. For each query, the size of both the query and the answer was first computed, when serialized in EXI4JSON, and then was recorded

5 http://swat.cse.lehigh.edu/projects/lubm/

	$\lvert\Omega\rvert$	QUERY (BYTES)	ANSWER (BYTES)
Q1	1	108	165
Q2	0	98	0
Q3	4	112	447
Q4	0*	137	0
Q5	32	91	2458
Q6	26	43	1826
Q7	15	138	1652
Q8	0*	116	0
Q9	3	100	603
Q10	1	108	165
Q11	1	79	119
Q12	0*	101	0
Q13	0*	84	0
Q14	18	43	1304

Table 11: Size of LUBM queries (frames) and answers (substitutions) serialized in EXI4JSON on a small synthetic dataset; because of its small size, the LUBM generator omits assertions in the dataset such that certain queries (marked with *) have no answer

the number of intermediate results (i.e. of mappings) during the execution of Alg. 2. The hypothesis is that all these quantities be reasonably lower than the total amount of RAM available in the chip.

Table 11 shows the size of serialized queries and answers. Queries are JSON-LD frames as per Def. 16 while answers are obtained by applying mappings returned by MATCH to an input frame. More precisely, answers correspond to the union of all "contextual" substitutions for the input frame (Def. 6). Both queries and answers are therefore JSON-LD graphs and can be serialized as described in earlier experiments in Sec. 4.4.1. LUBM includes an OWL ontology, which was used to generate a minimal context. As suggested by previous results, EXI4JSON was taken as serialization format.

The first thing we can observe on Table 11 is that all queries take a reasonable amount of RAM ($\leqslant$ 200B) in comparison to typical RAM resources on MCUs (8 to 64kB), which means that the entire frame can be loaded into memory before the algorithm starts. With this observation, we empirically confirm the relevance of *data* complexity analysis, which assumes that queries are bounded in size.

The size of the answers for e.g. Q5 is high and serializing it entirely in RAM would not always be feasible. If these answers are to be exchanged between constrained servients, a streaming approach must

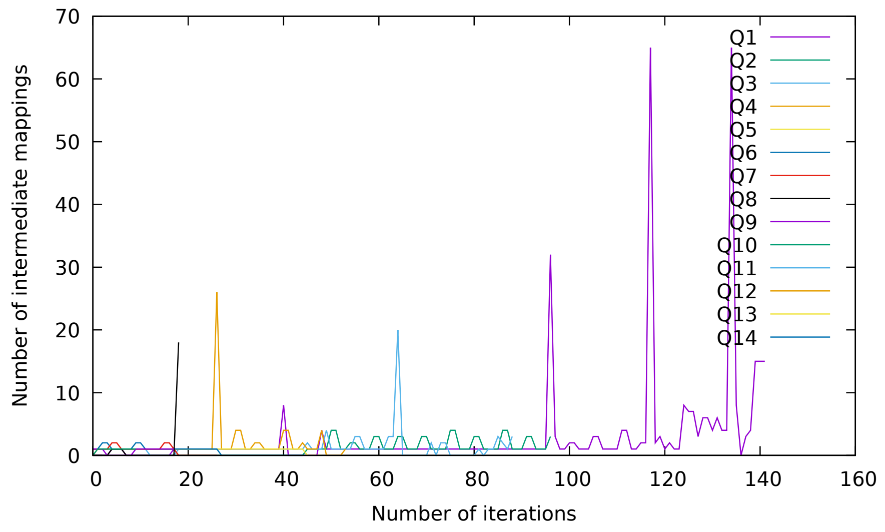


Figure 18: Evolution of the number of intermediate results over the execution of Alg. 2 for LUBM queries on a small synthetic dataset

be taken, like CoAP block-wise transfer [14]. However, throughout the execution of frame matching, a high number of mappings may be necessary to output the correct answer. This is the case with Q9 which contains three variables. To study the feasibility of complex frame matching, we should have a look at the evolution of the number of intermediate mappings generated by Alg. 2. The result for all LUBM queries is shown on Fig. 18. What comes out of this figure are sharp drops, particularly on Q9, which correspond to mappings generated by MATCHEXPRESSION that are filtered out when joining with incompatible mappings. While processing Q9, there can be up to three times more intermediate results than actual answers. In the worst case, MATCHEXPRESSION can generate $|N_\mathcal{G}|^{|V_\mathcal{F}|}$ mappings, where $|N_\mathcal{G}|$ is the number of IRIs in the input graph and $|V_\mathcal{F}|$ the number of variables in the input frame $\mathcal{F}$. All these mappings must be temporary stored in RAM. Given that LUBM queries have at most three variables, this upper bound for the present benchmarked dataset is of $196^3 = 7{,}529{,}536$ mappings. As Fig. 18 shows, actual numbers do not follow this exponential relation between graph, query and answers. This benchmark suggests that frame matching, and thus a certain level of autonomy in terms of knowledge management, is generally doable on MCUs.

My original implementation of the µRDF store was in the C programming language and targeted specifically the ESP8266 platform [9, 22]. It was demonstrated at ESWC 2017 in the form of a chat application with the MCU [22]. With this implementaton, frame matching terminates in less than 10ms for all queries of the LUBM query mix. As a comparison, the CoAP Round-Trip Time (RTT) for these datasets on the ESP8266 is of several orders of magnitude higher. RTT is de-

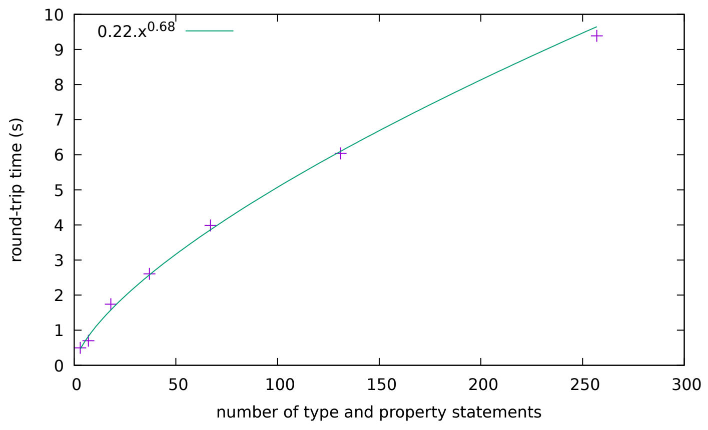


Figure 19: Evolution of the CoAP Round-trip Time over the number of ABox assertions in an LUBM dataset, serialized in EXI4JSON

fined as the total elapsed time starting when the first bytes of data are sent by the server until the client acknowledges reception of the whole dataset. To see the evolution of the CoAP RTT over the payload size, LUBM datasets of various sizes were generated. Results are shown on Fig. 19. We can observe a polynomial pattern with respect to the size of the dataset (from which directly depends the size in bytes of the message) but we can also see that all RTTs are above 100 ms, several orders of magnitude more than execution times to match frames locally. What follows from this observation is that LDP is no reasonable alternative to local querying on MCUs. It would indeed require to exchange all intermediate results over CoAP: query processing time would be generally high, hence high power consumption.

This observation also motivated the implementation of a second version of the µRDF store which would be more portable yet less performant. This second implementation, available online[6], complies to the ECMAScript 5.1 specification (a dialect of JavaScript) and can therefore be executed on MCUs via the JerryScript runtime[7]. Fig. 18 was compiled using this second version of the µRDF store.

4.5 SUMMARY

In this chapter, we got interested in the feasibility of exchanging TD documents in embedded environments, typically composed of MCUs interconnected in an IP network. It was first reported on the ongoing effort to port Web technologies to such environments, what is called the *Embedded Web* in the literature. These technologies include e.g. CoAP, EXI and CBOR, respectively inspired by HTTP, XML and

6 https://github.com/vcharpenay/uRDF.js
7 http://jerryscript.net

JSON. Binary formats for RDF have also been proposed, like HDT and RDF/EXI. Was then introduced the µRDF store, an embedded RDF database designed for MCUs. WoT servients can store locally TD documents in a µRDF store, update them and run queries against them.

My implementation of the µRDF store is based on a JSON format for RDF standardized by the W3C, called JSON-LD. The chapter introduced a new formalism for JSON-LD compaction and framing, which can be used to efficiently exchange and query RDF documents. On the one hand, JSON-LD compaction consists in mapping class, property and individual names to arbitrary UTF-8 strings as defined in a *context*. If a context is globally defined, any JSON-LD document can be sent in its compacted form, without context. On the other hand, JSON-LD framing is an alternative to SPARQL that can be directly performed on compacted documents. In Alg. 2, the classical SPARQL pattern matching algorithm was re-written in JSON-LD terms.

It was experimentally showed that JSON-LD compaction is more efficient than HDT and RDF/EXI in compacting TD documents, when binary formats for JSON are used (EXI4JSON, CBOR). Compaction was tested it on two datasets, taken from the SPITFIRE research project (weather stations) and the Siemens Desigo CC integration platform for BA systems. When all TD documents available in SPITFIRE of Desigo are merged, HDT remains the most efficient but when they are taken individually, JSON-LD compaction with EXI4JSON proved the most efficient, in average. The semantic discovery experiments of Sec. 3.4 was then reconsidered, to then apply EXI4JSON on the assertions being exchanged (TD documents and update ABoxes). All payloads can be sent in a single CoAP block, that is, efficiently between MCUs and a TDir. Finally, JSON-LD framing was tested on the LUBM SPARQL benchmark, a synthetic benchmark dealing with university datasets. The results we obtain from this benchmark indicate that frame matching can be implemented on MCUs for a reasonable cost in terms of RAM.

My µRDF store implementation was meant to be used in highly decentralized WoT systems in which servients are self-aware MCUs that expose their capabilities (as TD documents) and organize by themselves: a servient initiates an interaction by querying its local µRDF store to find suitable peers and then by following the hyperlinks these peers expose. This way, each servient becomes a knowledge-base intelligent system capable of autonomy. Servients are also adaptive insofar as servient interactions change as soon as the knowledge stored in their own µRDF store instance is updated. An example of update was presented in Ch. 3 in the context of semantic discovery (Def. 9). One can also think of arbitrary updates whenever the context of a servient changes (e.g. reconfiguration of a product line or new user preferences in a building).

5 CONCLUSION & PERSPECTIVES

5.1 THEORETICAL & EXPERIMENTAL RESULTS

The problem of this thesis was stated as the definition of formal semantics for WoT given a graph model for interactions between WoT servients. The general assumption was that knowledge graphs are good foundations for this purpose: an edge in the graph of interactions of a WoT system shall map to a path in a knowledge graph describing the components of the system and their environment. The most important aspects of the thesis can be summarized as follows:

- The W3C TD model proves a satisfactory basis to model 'things' and their properties when aligned with the SSN ontology, another W3C standard. It was quantitatively showed that the TD model, close to that of oneM2M, is compatible with most IoT and industrial protocols (like BLE, BACnet or OPC-UA) with respect to their data model. Moreover, SSN was showed to have become the reference for Web ontologies used in the IoT. The semantic alignment between the TD model and SSN therefore captures most of both worlds. On this basis, an extensive and consistent vocabulary to describe 'things' is available to annotate TD documents and build knowledge graphs for WoT.
- The semantics of a WoT system were defined as the annotation of edges in its graph of interactions with assertions found in a knowledge graph describing the system. On this basis, one can define various tasks like semantic discovery. A peculiarity of knowledge graphs in WoT is that they likely include anonymous entities, whose existence can be stated but that cannot be fully identified. The thesis contributes to the state-of-the-art in query answering on such knowledge graphs by providing a tractable skolemization algorithm that is sound and complete for a well-defined fragment of OWL.
- At last, it was verified that WoT knowledge graphs can be constructed for any system, including systems composed exclusively of constrained devices communicating in a peer-to-peer fashion. For this purpose, is introduced a formalization of JSON-LD, a serialization format for RDF that defines procedures for compaction and framing (similar to SPARQL querying). It was experimentally showed that JSON-LD compaction, when combined with a binary JSON serialization like EXI4JSON and CBOR, is more efficient than the state-of-the-art. JSON-LD framing (on compacted documents) was also tested on a querying

benchmark, showing that the procedure can reasonably be implemented on constrained devices like MCUs, paving the way to intelligent systems based on knowledge graphs and capable of autonomy and adaptibility.

As mentioned in introduction, the thesis refers to knowledge graphs from the point of view of standardized Semantic Web technologies (RDF, SPARQL, OWL, JSON-LD). Standards designed for the Embedded Web were also reused (EXI, CBOR). The use cases chosen for experiments (Intel labs, water management) were specifically chosen to be realistic enough to consider further implementation into product quality software. In other words, the outcomes of this thesis come with a certain level of technological maturity, which allows for a quick technology transfer. In fact, parts of its conclusions are to be found in the online documentation of the W3C TD ontology, which also provides guidelines on integrating Semantic Web technologies with its WoT framework.

Nonetheless, the use of knowledge graphs for WoT, as is being advocating for in this thesis, relies on certain assumptions that deserve closer evaluation. The following section revisits these assumptions and suggests future work to evaluate their relevance. It only mentions issues specific to WoT, as opposed to limitations inherent to knowledge graphs themeselves (they are e.g. prone to include inconsistencies or do not fit well in the presence of temporal data).

5.2 FUTURE WORK

Several assumptions were made throughout this work. Its three main assumptions are that knowledge graphs be the best representation for WoT servients and their environment (Sec. 1.2.2), that objects be a universal data structure to model the physical world (Sec. 2.2.1) and that an extensive Web ontology for physical bodies and their properties could be engineered with reasonable effort (Sec. 3.3.1). I now provide ideas for future work on these three assumptions.

First, in this thesis, knowledge graphs eclipsed alternative representations like numeric models and simluations. For instance, the graph of adjacencies describing the Intel Labs office (Sec. 3.4.1) would typically be provided in a BIM as a 3D model. Such a spatial model can also be used for thermic simulation in a so-called *digital twin* of the building [42]. Cause-and-effect reasoning can be implemented by comparing sensor measurements with their simulated counterpart on a digital twin. It is however not trivial to combine numeric models with symbolic ones like knowledge graphs to obtain realistic simulations. Digital twins would be a good application for research on this aspect.

Second, if objects are the most widespread atomic structure in computing, there is no evidence that they are the most accurate to model physical world entities [112]. In fact, there are discrepancies between objects used for pure modeling and the objects computer programs manipulate. The latter are often designed without considering whether they can be interpreted in the physical world or not. Conversely, it is notoriously arduous to program against Web ontologies, despite their object-oriented DL foundations. There has been little research on this duality, except to aknowledge it. The W3C WoT framework includes a standardized API for TD documents (similar to the Document Object Model for XML), which encourages research on programmatic aspects of Web ontologies.

Third, after observing that only few Web ontologies included existential restrictions, was identified the need for an ontology for physical bodies that includes such restrictions. Addressing this need is one of the most immediate avenues for future work and it may also be one of the easiest. This intuition is mainly based on two elements. First, SWEET, QUDT and OM, three ontologies in the field of physics, have been among the first extensive Web ontologies to be available on the Semantic Web: they represent a solid basis to start this work. Second, many simple laws of physics can be provided as logical rules, like the fact two miscible fluids have the same volume. It is also possible to derive inequalities between numeric values from boolean properties, like the fact a lamp is off also means that brightness it the corresponding space is low. The best way to validate this assumption is to start working on such an ontology with a strict knowledge engineering methodology and measure to which extent the rules of Sec. 3.3.1 can be integrated in a larger ontology.

5.3 FINAL WORD: THE LIFECYCLE OF A REVOLUTION

I mentioned in preamble of this thesis that it was mostly a "prototype-driven" work. It was indeed motivated by the desire to develop IoT and WoT technologies that could be integrated into existing industrial processes in a near future. I therefore dedicate the final word of this thesis to the prospect of a new form of industrialization with WoT and its consequences.

The title of this section was chosen after a presentation Jennifer Granick made at the 2015 Black Hat Briefings in Las Vegas, where she questioned whether the Internet will remain the same open and unregulated—in one word, *revolutionary*—cyberspace as in the 90' after the manufacturing industry settles in it. As a lawyer, she points at the effects of the IoT on laws for computing and summarizes: today, Oracle is not being sued for failures while Chrysler is. Information Technology companies must indeed expect to have to deal with software liability issues when autonomous cars embed their data analysis systems and when building operators or households make use of their intelligent assistants.

When it comes to software liability, Stuxnet, again, is a good example of what the future of WoT could look like. A deactivation date was hard-coded in Stuxnet's program after which it would stop spreading if it receives no command. This date, January 11th, 2009, corresponds to the end of the last presidential term of George W. Bush, whom any intelligence service in the United States was subordinated at the time. By law, the creators of Stuxnet had to include in its design a red security button that could stop all activities of the malicious program. Stuxnet had a precise log of its activity as well, at the disposal of decision makers, which turned out to help computer security experts decypher Stuxnet's peregrinations. By design, Stuxnet was transparent—at least, to its owners.

In her Black Hat keynote, Granick predicts the end of the Internet as we know it and the rise of a world of pervasive computing backed by monopolistic economic structures, which comes with serious privacy violation threats. In contrast to this vision, the W3C WoT framework aims at introducing openness and decentralization in industrial systems. The semantic framework for WoT developed in this thesis goes in the same direction insofar as it defines WoT systems as multi-agent systems in which agents are capable of autonomy: no central authority can have full control over a WoT system, unlike Stuxnet. At first sight, decentralization complexifies the implementation of necessary security measures. Yet, a formalization of the semantics of WoT brings certain guarantees for its industrialization.

The first guarantee provided by WoT semantics is determinism. In any cyber-physical system, there is indeed a coupling between the level of perception of the physical world and expectations, that is, its prior knowledge of it. In formalizing the relation between the two as was done in this thesis, it is possible to fully determine one from the other. For instance, the set of possible graphs of interactions in a system is fixed as soon as the set of ontologies used to model it is also fixed. This set can be e.g. defined in terms of alignments in a cloud of Web ontologies, as the one constructed throughout the thesis.

The second guarantee of WoT semantics is that no assertion is inferred from the *absence* of information. It is likely that WoT agents will tend to discard what they cannot perceive, a well-known cognitive bias theoretized by Daniel Kahneman as 'what you see is all there is'. Reasoning with a "closed world", that is, with the assumption that whatever is not asserted is false, can lead to potentially harmful miscalculations. For instance, an autonomous car should not infer from the fact it does not detect anything on the road that there is indeed no obstacle. Similarly, if one of the centrifuges in Natanz crashes under the action of Stuxnet, it should not follow that it is not physically there anymore. Going further, existential reasoning, as is exposed in Ch. 3, provides a mean to distinguish between the absence of sensing and sensor failure, whenever a knowledge base states that an entity *should* exist. In the case of Natanz, if the centrifuge is digitally tagged, a WoT system should be able to detect a failure if no speed is measured: every centrifuge does have a rotor speed.

For all these reasons, I strongly believe that industrial systems would benefit from adopting WoT technologies. WoT will likely not be a revolution in human history but we have not yet come to the end of the Internet and the Web.

BIBLIOGRAPHY

[1] Kevin Ashton. “That 'Internet of Things' Thing.” In: *RFID Journal* (June 22, 2009). URL: http://www.rfidjournal.com/articles/view?4986.

[2] Franz Baader, Sebastian Brandt, and Carsten Lutz. “Pushing the EL envelope.” In: *roceedings of the Nineteenth International Joint Conference on Artificial Intelligence, IJCAI-05*. Vol. 5. International Joint Conferences on Artificial Intelligence Organization, 2005, pp. 364–369.

[3] Franz Baader, Sebastian Brandt, and Carsten Lutz. “Pushing the EL envelope further.” In: *Proceedings of the Fourth OWLED Workshop on OWL: Experiences and Directions*. CEUR-WS, 2008.

[4] Franz Baader, Diego Calvanese, Deborah L. McGuiness, Daniele Nardi, and Peter Patel-Schneider, eds. *The description logic handbook: theory, implementation and applications*. English. Cambridge: Cambridge University Press, 2010. ISBN: 978-0-511-71178-7.

[5] Sebastian R. Bader, Tobias Käfer, Lars Heling, Raphael Manke, and Andreas Harth. “Exposing Internet of Things Devices via REST and Linked Data Interfaces.” In: *Proceedings of the 2nd Workshop Semantic Web Technologies for the Internet of Things*. CEUR-WS, 2017.

[6] Bharathan Balaji et al. “Brick: Towards a Unified Metadata Schema For Buildings.” In: *Proceedings of the 3rd ACM International Conference on Systems for Energy-Efficient Built Environments*. BuildSys '16. Palo Alto, CA, USA: ACM, 2016, pp. 41–50. ISBN: 978-1-4503-4264-3.

[7] Steve Battle and Steve Speicher. *Linked Data Platform Use Cases and Requirements*. Mar. 13, 2014. URL: https://www.w3.org/TR/ldp-ucr/.

[8] Joel Bender. *BACnet Ontology*. 2012. URL: http://bacowl.sourceforge.net/.

[9] Florian Bieringer. “Querying RDF Data in a Constrained Environment with Focus on M2M Interaction.” Master's Thesis. Passau: Universität Passau, Mar. 2017.

[10] Christian Bizer and Andreas Schultz. “The Berlin SPARQL Benchmark.” In: *Semantic Services, Interoperability and Web Applications: Emerging Concepts*. IGI Global, 2011, pp. 81–103.

[11] Dario Bonino and Fulvio Corno. “DogOnt - Ontology Modeling for Intelligent Domotic Environments.” In: *The Semantic Web - ISWC 2008*. Ed. by Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan. Vol. 5318. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 790–803.

[12] Carsten Bormann, Mehmet Ersue, and Ari Keränen. *Terminology for Constrained-Node Networks*. RFC 7228. May 2014. URL: `https://rfc-editor.org/rfc/rfc7228.txt`.

[13] Carsten Bormann and Paul E. Hoffman. *Concise Binary Object Representation (CBOR)*. RFC 7049. Oct. 2013. URL: `https://rfc-editor.org/rfc/rfc7049.txt`.

[14] Carsten Bormann and Zach Shelby. *Block-Wise Transfers in the Constrained Application Protocol (CoAP)*. RFC 7959. Aug. 2016. URL: `https://rfc-editor.org/rfc/rfc7959.txt`.

[15] Dan Brickley and Libby Miller. *FOAF vocabulary specification*. 2010. URL: `http://xmlns.com/foaf/spec/`.

[16] Sergey Brin and Lawrence Page. “The anatomy of a large-scale hypertextual Web search engine.” In: *Proceedings of the Seventh International World Wide Web Conference*. Vol. 30. 1. 1998, pp. 107 –117.

[17] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. “Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family.” In: *Journal of Automated Reasoning* 39.3 (Oct. 2007), pp. 385–429. ISSN: 0168-7433, 1573-0670.

[18] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. “Data complexity of query answering in description logics.” In: *Artificial Intelligence* 195 (2013), pp. 335 –360. ISSN: 0004-3702.

[19] Andrea Calì, Georg Gottlob, and Thomas Lukasiewicz. “A general Datalog-based framework for tractable query answering over ontologies.” In: *Web Semantics: Science, Services and Agents on the World Wide Web* 14 (July 2012), pp. 57–83. ISSN: 15708268.

[20] S. Ceri, G. Gottlob, and L. Tanca. “What you always wanted to know about Datalog (and never dared to ask).” In: *IEEE Transactions on Knowledge and Data Engineering* 1.1 (Mar. 1989), pp. 146–166. ISSN: 10414347.

[21] V. Charpenay, S. Käbisch, D. Anicic, and H. Kosch. “An ontology design pattern for IoT device tagging systems.” In: *2015 5th International Conference on the Internet of Things (IOT)*. Oct. 2015, pp. 138–145.

[22] Victor Charpenay, Sebastian Käbisch, and Harald Kosch. "μRDF Store: Towards Extending the Semantic Web to Embedded Devices." In: *The Semantic Web: ESWC 2017 Satellite Events*. Vol. 10577. Cham: Springer International Publishing, 2017, pp. 76–80.

[23] Victor Charpenay, Sebastian Käbisch, and Harald Kosch. "Towards a Binary Object Notation for RDF." In: *The Semantic Web*. Springer International Publishing, 2018, pp. 97–111. ISBN: 978-3-319-93417-4.

[24] Victor Charpenay, Sebastian Kaebisch, and Harald Kosch. "Introducing Thing Descriptions and Interactions: An Ontology for the Web of Things." In: *Joint Proceedings of the 3rd Stream Reasoning (SR 2016) and the 1st Semantic Web Technologies for the Internet of Things (SWIT 2016) workshops*. Semantic Web Technologies for the Internet of Things 2016. Kobe: CEUR-WS, Oct. 17, 2016.

[25] Victor Charpenay, Sebastian Kaebisch, and Harald Kosch. "A Framework for Semantic Discovery on the Web of Things." In: *Studies on the Semantic Web* (2018), pp. 147–162. ISSN: 1868-1158.

[26] Victor Charpenay, Sebastian Käbisch, and Harald Kosch. "Semantic data integration on the web of things." In: *Proceedings of the 8th International Conference on the Internet of Things - IOT '18*. the 8th International Conference. Santa Barbara, California: ACM Press, 2018, pp. 1–8. ISBN: 978-1-4503-6564-2.

[27] Alonzo Church. "A formulation of the logic of sense and denotation." In: *Structure, method, and meaning: Essays in honor of Henry M. Sheffer* (1951), pp. 3–24. ISSN: 0022-4812, 1943-5886.

[28] 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: 1570-8268.

[29] Jianfeng Du, Kewen Wang, and Yi-Dong Shen. "A Tractable Approach to Abox Abduction over Description Logic Ontologies." In: *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*. AAAI'14. Québec City, Québec, Canada: AAAI Press, 2014, pp. 1034–1040.

[30] Adam Dunkels. "Programming Memory-Constrained Networked Embedded Systems." PhD thesis. Stokholm: Swedish Institute of Computer Science, Feb. 2007.

[31] Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007. ISBN: 978-3-540-49611-3.

[32] Christina Feier, Axel Polleres, Roman Dumitru, John Domingue, Michael Stollberg, and Dieter Fensel. "Towards intelligent web services: the web service modeling ontology (WSMO)." In: *2005 International Conference on Intelligent Computing (ICIC'05)*. 2005.

[33] Christiane Fellbaum, ed. *WordNet : an electronic lexical database*. Cambridge, Massachusetts: MIT Press, 1998. ISBN: 9780262061971.

[34] Javier D. Fernández, Alejandro Llaves, and Oscar Corcho. "Efficient RDF Interchange (ERI) Format for RDF Data Streams." In: *The Semantic Web – ISWC 2014*. Vol. 8797. Cham: Springer International Publishing, 2014, pp. 244–259.

[35] Javier D. Fernández, Miguel A. Martínez-Prieto, and Claudio Gutierrez. "Compact Representation of Large RDF Data Sets for Publishing and Exchange." In: *The Semantic Web – ISWC 2010*. Vol. 6496. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 193–208.

[36] Alexander R Galloway and Eugene Thacker. *The exploit: A theory of networks*. University of Minnesota Press, 2007. 208 pp. ISBN: 978-0-8166-5044-6.

[37] Aldo Gangemi and Valentina Presutti. "The bourne identity of a web resource." In: *Proceedings of Identity Reference and the Web Workshop (IRW) at the WWW Conference*. WWW 2006. Edinburgh, May 2006.

[38] Sandra Álvarez García, Nieves R. Brisaboa, Javier D. Fernández, and Miguel A. Martínez-Prieto. "Compressed k^2-Triples for Full-In-Memory RDF Engines." In: *AMCIS 2011 Proceedings – All submissions*. Vol. 350. 2011.

[39] Francis Gasse, Ulrike Sattler, and Volker Haarslev. "Rewriting Rules into SROIQ Axioms." In: *Proceedings of the DL 21st International Workshop on Description Logics (DL2008)*. Vol. 353. Dresen, Germany: CEUR-WS, May 13, 2008.

[40] Marco Gavanelli, Evelina Lamma, Fabrizio Riguzzi, Elena Bellodi, Riccardo Zese, and Giuseppe Cota. "Reasoning on Datalog$^{\pm}$ Ontologies with Abductive Logic Programming." In: *Fundamenta Informaticae* 1 (Mar. 7, 2018), pp. 65–93.

[41] Alex Gibney. *Zero Days*. United States of America, 2016.

[42] Edward Glaessgen and David Stargel. "The digital twin paradigm for future NASA and US Air Force vehicles." In: *53rd AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics and Materials Conference 20th AIAA/ASME/AHS Adaptive Structures Conference 14th AIAA*. 2012, p. 1818.

[43] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. "Description Logic Programs: Combining Logic Programs with Description Logic." In: *Proceedings of the 12th International Conference on World Wide Web*. WWW '03. Budapest, Hungary: ACM, 2003, pp. 48–57. ISBN: 1-58113-680-3.

[44] The W3C SPARQL Working Group. *SPARQL 1.1 Overview*. Mar. 21, 2013. URL: https://www.w3.org/TR/sparql11-overview/.

[45] Thomas R. Gruber. "Toward principles for the design of ontologies used for knowledge sharing?" In: *International Journal of Human-Computer Studies* 43.5 (1995), pp. 907 –928. ISSN: 1071-5819.

[46] Dominique Guinard and Vlad M Trifa. "Towards the Web of Things: Web Mashups for Embedded Devices." In: *Proceedings of WWW (International World Wide Web Conferences)*. Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009). Madrid, Spain, Apr. 2009.

[47] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. "LUBM: A benchmark for OWL knowledge base systems." In: *Web Semantics: Science, Services and Agents on the World Wide Web* 3.2 (Oct. 2005), pp. 158–182. ISSN: 15708268.

[48] Claudio Gutierrez, Carlos Hurtado, and Alberto O. Mendelzon. "Foundations of semantic web databases." In: ACM Press, 2004, p. 95.

[49] A. Gyrard, C. Bonnet, K. Boudaoud, and M. Serrano. "LOV4IoT: A Second Life for Ontology-Based Domain Knowledge to Build Semantic Web of Things Applications." In: *2016 IEEE 4th International Conference on Future Internet of Things and Cloud (FiCloud)*. Aug. 2016, pp. 254–261.

[50] Amelie Gyrard, Soumya Kanti Datta, Christian Bonnet, and Karima Boudaoud. "Cross-Domain Internet of Things Application Development: M3 Framework and Evaluation." In: IEEE, Aug. 2015, pp. 9–16. ISBN: 978-1-4673-8103-1.

[51] Armin Haller, Krzysztof Janowicz, Simon Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. *Semantic Sensor Network Ontology*. W3C Recommendation. Oct. 19, 2017. URL: https://www.w3.org/TR/vocab-ssn/.

[52] H. Hasemann, A. Kroller, and M. Pagel. "RDF provisioning for the Internet of Things." In: Internet of Things (IOT), 2012 3rd International Conference on the. 2012, pp. 143–150.

[53] Henning Hasemann, Alexander Kröller, and Max Pagel. "The Wiselib TupleStore: A Modular RDF Database for the Internet of Things." In: (Mar. 5, 2014). URL: http://arxiv.org/abs/1402.7228.

[54] Patrick J. Hayes and Peter Patel-Schneider. *RDF 1.1 Semantics*. URL: http://www.w3.org/TR/rdf11-mt/.

[55] J. Hendler. "Agents and the Semantic Web." In: *IEEE Intelligent Systems* 16.2 (Mar. 2001), pp. 30–37.

[56] Martin Hepp and Andreas Radinger. *eClassOWL - The Web Ontology for Products and Services*. 2010. URL: http://www.heppnetz.de/projects/eclassowl/.

[57] Jack Hodges, Kimberly García, and Steven Ray. "Semantic Development and Integration of Standards for Adoption and Interoperability." In: *IEEE Computer* (2017).

[58] Aidan Hogan. "Skolemising Blank Nodes While Preserving Isomorphism." In: *Proceedings of the 24th International Conference on World Wide Web*. WWW '15. Florence, Italy: International World Wide Web Conferences Steering Committee, 2015, pp. 430–440.

[59] Aidan Hogan. "Canonical Forms for Isomorphic and Equivalent RDF Graphs: Algorithms for Leaning and Labelling Blank Nodes." In: *ACM Trans. Web* 11.4 (July 2017), 22:1–22:62.

[60] Ian Jacobs and Norman Walsh. *Architecture of the World Wide Web, Volume One*. W3C Recommendation. Dec. 15, 2004. URL: https://www.w3.org/TR/2004/REC-webarch-20041215/.

[61] Sebastian Kaebisch and Takuki Kamiya. *Web of Things (WoT) Thing Description*. W3C First Public Working Draft. Sept. 14, 2017. URL: https://www.w3.org/TR/wot-thing-description/.

[62] A. C. Kakas, R. A. Kowalski, and F. Toni. "Abductive Logic Programming." In: *Journal of Logic and Computation* 2 (1992), pp. 719–770.

[63] Aqeel Kazmi, Zeeshan Jan, Achille Zappa, and Martin Serrano. "Overcoming the Heterogeneity in the Internet of Things for Smart Cities." In: *Interoperability and Open-Source Solutions for the Internet of Things*. Cham: Springer International Publishing, 2017, pp. 20–35.

[64] Kajimoto Kazuo, Matthias Kovatsch, and Uday Davuluru. *Web of Things (WoT) Architecture*. W3C First Public Working Draft. Sept. 14, 2017. URL: https://www.w3.org/TR/wot-architecture/.

[65] Michael Kifer, Georg Lausen, and James Wu. "Logical foundations of object-oriented and frame-based languages." In: *Journal of the ACM* 42.4 (July 1, 1995), pp. 741–843.

[66] Szymon Klarman, Ulle Endriss, and Stefan Schlobach. "ABox Abduction in the Description Logic $\mathcal{ALC}$." In: *Journal of Automated Reasoning* 46 (Feb. 23, 2010), pp. 43–80.

[67] Markus Krötzsch. *Description Logic Rules*. Vol. 008. Studies on the Semantic Web. IOS Press/AKA, 2010. ISBN: 978-1-60750-654-6.

[68] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. "Conjunctive Queries for a Tractable Fragment of OWL 1.1." In: *The Semantic Web*. Vol. 4825. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 310–323.

[69] Markus Krötzsch, Sebastian Rudolph, and Pascal Hitzler. "ELP: Tractable Rules for OWL 2." In: *The Semantic Web - ISWC 2008*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 649–664.

[70] Sebastian Käbisch. "Resource Optimization of SOA-Technologies in Embedded Networks." PhD thesis. Universität Passau, 2014.

[71] Sebastian Käbisch, Daniel Peintner, and Darko Anicic. "Standardized and Efficient RDF Encoding for Constrained Embedded Networks." In: *The Semantic Web. Latest Advances and New Domains*. Vol. 9088. Cham: Springer International Publishing, 2015, pp. 437–452. (Visited on 04/29/2017).

[72] Tobias Käfer and Andreas Harth. "Rule-based Programming of User Agents for Linked Data." In: *Proceedings of the 11th International Workshop on Linked Data on the Web at the Web Conference (27th WWW)*. CEUR-WS, Apr. 2018.

[73] Markus Lanthaler, Michael Granitzer, and Christian Gütl. "Semantic Web Services: State of the Art." In: *Proceeding of the IADIS International Conference on Internet Technologies and Society 2010*. 2010.

[74] Danh Le Phuoc and Manfred Hauswirth. "Linked Open Data in Sensor Data Mashups." In: *Proceedings of the 2nd International Workshop on Semantic Sensor Networks*. Vol. 522. Washington DC: CEUR-WS, 2009.

[75] Anh Le-Tuan, Conor Hayes, Marcin Wylot, and Danh Le-Phuoc. "RDF4Led: an RDF engine for lightweight edge devices." In: *Proceedings of the 8th International Conference on the Internet of Things - IOT '18*. Santa Barbara, California: ACM Press, 2018, pp. 1–8.

[76] Edward A Lee. "Cyber physical systems: Design challenges." In: *11th IEEE Symposium on Object Oriented Real-Time Distributed Computing (ISORC)*. IEEE, 2008, pp. 363–369.

[77] Maxime Lefrançois. "Planned ETSI SAREF Extensions based on the W3C&OGC SOSA/SSN-compatible SEAS Ontology Pattern." In: SIS-IoT: Semantic Interoperability and Standardization in the IoT. CEUR-WS, 2017.

[78] Maxime Lefrançois, Jarmo Kalaoja, Takoua Guariani, and Antoine Zimmermann. *SEAS Knowledge Model*. Deliverable 2.2. ITEA2 12004 Smart Energy Aware Systems, 2016.

[79] Jacqueline Leighton and Robert Sternberg. *The nature of reasoning*. Cambridge, U.K. New York: Cambridge University Press, 2004. ISBN: 978-0-521-00928-7.

[80] P. Levis et al. "TinyOS: An Operating System for Sensor Networks." In: *Ambient Intelligence*. Berlin/Heidelberg: Springer-Verlag, 2005, pp. 115–148.

[81] Steffen Lohmann, Stefan Negru, Florian Haag, and Thomas Ertl. "Visualizing Ontologies with VOWL." In: *Semantic Web* 7.4 (2016), pp. 399–419.

[82] Dave Longley, Gregg Kellogg, Markus Lanthaler, and Manu Sporny. *JSON-LD 1.0 Processing Algorithms and API*. W3C Recommendation. Jan. 16, 2014. URL: https://www.w3.org/TR/json-ld-api/.

[83] Giuseppe Loseto, Saverio Ieva, Filippo Gramegna, Michele Ruta, Floriano Scioscia, and Eugenio Di Sciascio. "Linked Data (in low-resource) Platforms: a mapping for Constrained Application Protocol." In: *Proceedings of 15th International Semantic Web Conference (ISWC 2016)*. International Semantic Web Conference. Kobe, Oct. 2016.

[84] Samuel R. Madden. "The design and evaluation of a query processing architecture for sensor networks." PhD thesis. Massachusetts Institute of Technology, 2003.

[85] David Martin et al. *OWL-S: Semantic Markup for Web Services*. Nov. 22, 2004. URL: http://www.w3.org/Submission/OWL-S/.

[86] Alistair Miles and Sean Bechhofer. *SKOS Simple Knowledge Organization System Reference*. W3C Recommendation. Aug. 18, 2009. URL: http://www.w3.org/TR/skos-reference.

[87] João Moreira, Laura Daniele, Luis Ferreira Pires, Marten van Sinderen, Katarzyna Wasielewska, Pawel Szmeja, Wiesław Pawłowski, Maria Ganzha, and Marcin Paprzycki. "Towards IoT platforms' integration: Semantic Translations between W3C SSN and ETSI SAREF." In: SIS-IoT: Semantic Interoperability and Standardization in the IoT. CEUR-WS, 2017.

[88] Yavor Nenov, Robert Piro, Boris Motik, Ian Horrocks, Zhe Wu, and Jay Banerjee. "RDFox: A Highly-Scalable RDF Store." In: *The Semantic Web - ISWC 2015*. Vol. 9367. Cham: Springer International Publishing, 2015, pp. 3–20.

[89] H. Michael Newman. "BACnet Explained, Part One." In: *Supplement to ASHRAE Journal* 55.11 (Nov. 2013), B2–B7.

[90] Pieter Pauwels and Walter Terkaj. “EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology.” In: *Automation in Construction* 63 (Mar. 2016), pp. 100–133.

[91] Carlos Pedrinaci, Dong Liu, Maria Maleshkova, David Lambert, Jacek Kopecky, and John Domingue. “iServe: a linked services publishing platform.” In: *Ontology Repositories and Editors for the Semantic Web Workshop at The 7th Extended Semantic Web*. Vol. 596. CEUR-WS, 2010.

[92] Daniel Peintner and Don Brutzman. *EXI for JSON (EXI4JSON)*. W3C Working Draft. Aug. 23, 2016. URL: https://www.w3.org/TR/exi-for-json/.

[93] D. Pfisterer et al. “SPITFIRE: toward a semantic web of things.” In: *IEEE Communications Magazine* 49.11 (Nov. 2011), pp. 40–48.

[94] Stefan Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. John Wiley & Sons, Inc., Mar. 31, 2009. ISBN: 978-0-470-77944-6.

[95] Valentina Presutti and Aldo Gangemi. “Identity of resources and entities on the web.” In: *Progressive Concepts for Semantic Web Evolution: Applications and Developments: Applications and Developments* (2010), pp. 123–147.

[96] *QUDT Ontologies Overview*. 2014. URL: http://www.qudt.org.

[97] Robert G. Raskin and Michael J. Pan. “Knowledge representation in the semantic web for Earth and environmental terminology (SWEET).” In: *Computers & Geosciences* 31.9 (2005), pp. 1119 –1125.

[98] Mads Holten Rasmussen, Pieter Pauwels, Maxime Lefrançois, Georg Ferdinand Schneider, Christian Anker Hviid, and Jan Karlshøj. “Recent changes in the Building Topology Ontology.” In: *LDAC2017 - 5th Linked Data in Architecture and Construction Workshop*. Dijon, France, Nov. 2017.

[99] Sebastian Rudolph. “Foundations of Description Logics.” In: *Reasoning Web. Semantic Technologies for the Web of Data*. Springer Berlin Heidelberg, Jan. 1, 2011, pp. 76–136.

[100] Stuart J. Russell and Peter Norvig. *Artificial intelligence: a modern approach*. In collab. with Ernest Davis. 3rd ed. Prentice Hall series in artificial intelligence. Upper Saddle River: Prentice Hall, 2010. 1132 pp. ISBN: 978-0-13-604259-4.

[101] Michele Ruta, Floriano Scioscia, Eugenio Di Sciascio, and Giuseppe Loseto. “Semantic-Based Enhancement of ISO/IEC 14543-3 EIB/KNX Standard for Building Automation.” In: *IEEE Transactions on Industrial Informatics* 7.4 (Nov. 2011), pp. 731–739.

[102] Konstantinos Sagonas, Terrance Swift, and David S. Warren. “XSB as an efficient deductive database engine.” In: *ACM SIGMOD Record* 23.2 (June 1, 1994), pp. 442–453.

[103] Sanjay Sarma, David L. Brock, and Kevin Ashton. *The Networked Physical World*. White Paper. Oct. 1, 2000.

[104] M. Satyanarayanan. “Pervasive computing: vision and challenges.” In: *IEEE Personal Communications* 8.4 (Aug. 2001), pp. 10–17.

[105] Leo Sauermann and Richard Cyganiak. *Cool URIs for the Semantic Web*. W3C Interest Group Note. W3C, Dec. 3, 2008. URL: https://www.w3.org/TR/cooluris/.

[106] John Schneider, Takuki Kamiya, Daniel Peintner, and Rumen Kyusakov. *Efficient XML Interchange (EXI) Format 1.0 (Second Edition)*. Feb. 11, 2014. URL: http://www.w3.org/TR/exi/.

[107] Nicolas Seydoux, Khalil Drira, Nathalie Hernandez, and Thierry Monteil. “Autonomy through knowledge: how IoT-O supports the management of a connected apartment.” In: *Joint Proceedings of the 3rd Stream Reasoning (SR 2016) and the 1st Semantic Web Technologies for the Internet of Things (SWIT 2016) workshops*. Semantic Web Technologies for the Internet of Things 2016. Kobe: CEUR-WS, Oct. 2016.

[108] Zach Shelby. “Embedded web services.” In: *IEEE Wireless Communications* 17.6 (Dec. 2010), pp. 52–57.

[109] Zach Shelby. *Constrained RESTful Environments (CoRE) Link Format*. RFC 6690. Aug. 2012. URL: https://rfc-editor.org/rfc/rfc6690.txt.

[110] Zach Shelby, Klaus Hartke, and Carsten Bormann. *The Constrained Application Protocol (CoAP)*. RFC 7252. June 2014. URL: https://rfc-editor.org/rfc/rfc7252.txt.

[111] *SmartM2M; Smart Appliances Reference Ontology and oneM2M Mapping*. Technical Specification ETSI TS 103 264. ETSI, Nov. 2015. (Visited on 10/14/2016).

[112] Brian Cantwell Smith. *On the origin of objects*. Cambridge, Massachusetts: MIT Press, 1996. 420 pp. ISBN: 978-0-262-19363-4.

[113] Steve Speicher, John Arwe, and Ashok Malhotra. *Linked Data Platform 1.0*. Feb. 26, 2015. URL: https://www.w3.org/TR/ldp/.

[114] Manu Sporny, Dave Longley, Gregg Kellogg, Markus Lanthaler, and Niklas Lindström. *JSON-LD 1.0 - A JSON-based Serialization for Linked Data*. W3C Recommendation. Jan. 16, 2014. URL: http://www.w3.org/TR/json-ld/.

[115] Manu Sporny, Gregg Kellogg, Dave Longley, and Markus Lanthaler. *JSON-LD Framing 1.1*. W3C Draft Community Group Report. Oct. 31, 2017.

[116] Aparna Saisree Thuluva, Kirill Dorofeev, Monika Wenger, Darko Anicic, and Sebastian Rudolph. “Semantic-Based Approach for Low-Effort Engineering of Automation Systems.” In: *On the Move to Meaningful Internet Systems. OTM 2017 Conferences*. Vol. 10574. Cham: Springer International Publishing, 2017, pp. 497–512.

[117] Pierre-Yves Vandenbussche, Ghislain A. Atemezing, María Poveda-Villalón, and Bernard Vatant. “Linked Open Vocabularies (LOV): a gateway to reusable semantic vocabularies on the Web.” In: *Semantic Web Journal* (2014).

[118] Bernard Vatant. *GeoNames Ontology*. 2012. URL: http://www.geonames.org/ontology.

[119] Zhe Wang, Mahsa Chitsaz, Kewen Wang, and Jianfeng Du. “Towards Scalable and Complete Query Explanation with OWL 2 EL Ontologies.” In: *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. CIKM ’15. Melbourne, Australia: ACM, 2015, pp. 743–752.

[120] M. Weiser. “Hot topics-ubiquitous computing.” In: *Computer* 26.10 (Oct. 1993), pp. 71–72.

[121] Norbert Wiener. *Cybernetics; Or, Control and Communication in the Animal and the Machine*. Actualités scientifiques et industrielles. Technology Press, 1948. ISBN: 978-0-262-73009-9.

[122] Eli Zelkha and Brian Epstein. “From Devices to ‘Ambient Intelligence’: The Transformation of Consumer Electronics.” Philips Digital Living Room Conference. 1998.

[123] *oneM2M Base Ontology*. oneM2M Technical Specification TS-0012-V-0.7.0. oneM2M, Jan. 30, 2016.

INDEX

µRDF Store, 9, 74, 80
k^2-triples, 77, 87
6LoWPAN, 75

Abduction, 51
ABox, 45
Ambiant Intelligence, 3

BLE, 4
BSBM, 92
Building Automation, 41

CBOR, 75, 87
CBox, 45
Closed World Assumption, 84
CoAP, 75, 94
Contiki, 75
CoRE Link Format, 75
CURIE, 15
Cyber-physical System, 3
Cybernetics, 3

Data Complexity, 50
Decidability, 46
Deductive Database, 64
Description Logics, 45

Embedded Web, 74
Entailment, 42
ERI, 79
ESP8266, 68, 91, 92, 94
EXI, 75, 87

F-Logic, 81
FOL, 45

Graph of Alignments, 22
Graph of Interactions, 7, 59

HDT, 76, 87
HVAC, 65
Hyperlink, 30

IETF, 75
Industrial Network, 1
Intelligent System, 73
Interaction Affordance, 7, 30
Internet of Things, 3
Interpretation, 16
IRE, 35
IRI, 14, 35

JSON-LD, 80

Knowledge Base, 16, 42, 45
Knowledge Graph, 6, 14

LDP, 80
LOV4IoT, 17
LUBM, 92

Micro-controller, 68, 73
Model theory, 46
Multi-Agent System, 7, 59

Ontology Alignment, 16, 22
Ontology Matching, *see also* Ontology Alignment
Open World Assumption, *see also* Closed World Assumption

Pervasive Computing, 3
Physical Web, 41
Physical World Entity, 4

Query Answering, 49
Query Complexity, *see also* Data Complexity
Query Explanation, 50

RDF, 6, 14, 43
RDF4Led, 79
RFID, 3, 41

Semantic Discovery, 57
Semantic Web, 14
Sensor Network, 3, 65

Servient, 7
Skolemization, 60
SPARQL, 63, 84
SPITFIRE, 76, 88
SSN, 31
Stuxnet, 1, 34

Thing, 4, 17, 26, 31
Thing Description, 6, 25, 30
Thing Directory, 9, 64
TinyOS, 75
Tractability, 47, 74
Turtle, 80

Ubiquitous Computing, 3, 73
UML, 23
URI, *see also* IRI
URN, 36

VOWL, 31

Web Architecture, 14
Web of Things, 2, 7, 74
Web Ontology, 6
Web resource, 14
Wiselib, 76
WordNet, 23

Zigbee, 4

COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both LaTeX and LyX:

https://bitbucket.org/amiede/classicthesis/

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

http://postcards.miede.de/

Final Version as of August 16, 2019 (`classicthesis` version 1.0).

A AN OBJECT NOTATION FOR WEB ONTOLOGIES

The Web Ontology Language (OWL) has its own functional syntax that maps to RDF. Throughout the thesis, we have used an alternative object-oriented syntax inspired by F-logic. In Table 12, we introduce a mapping to the RDF variant of OWL. Then, we quickly introduce the principles of transforming tree-shaped rules as defined in Def. 3 to OWL axioms. See Ex. 24.

OWL	OBJECT NOTATION
`:C a owl:Class .`	C
`:p a owl:ObjectProperty .`	p
`:a a owl:NamedIndividual .`	a
`[] a owl:Restriction ;` `owl:onProperty :p ;` `owl:someValuesFrom :C .`	$x[\mathsf{p} \Rightarrow \mathsf{C}]$
`[] a owl:Restriction ;` `owl:onProperty :p ;` `owl:allValuesFrom :C .`	$x{:}\mathsf{C} :- x[\mathsf{p} \rightarrow y]$
`[] a owl:Restriction ;` `owl:onProperty :p ;` `owl:onClass :C ;` `owl:cardinality "n" .`	$x[\mathsf{p} \twoheadrightarrow \{\ x_1, \ldots, x_n\ \}]$
`[] owl:complementOf :C .`	$\neg\mathsf{C}$
`:C rdfs:subClassOf :D .`	$x{:}\mathsf{D} :- x{:}\mathsf{C}.$
`:p a owl:TransitiveProperty .`	$x[\mathsf{p} \rightarrow z] :-$ $x[\mathsf{p} \rightarrow y]$ and $y[\mathsf{p} \rightarrow z]$.
`:p a owl:FunctionalProperty .`	$x = y :-$ $x[\mathsf{p} \rightarrow z]$ and $y[\mathsf{p} \rightarrow z]$.
`[] owl:inverseOf :p .`	p^{-}
`:p a owl:SymmetricProperty .`	$x[\mathsf{p} \rightarrow y] :- z[\mathsf{p} \rightarrow x].$
`:p a owl:ReflexiveProperty .`	$x[\mathsf{p} \rightarrow x].$
`:p owl:propertyChainAxiom` `(:q :r) .`	$x[\mathsf{p} \rightarrow z] :-$ $x[\mathsf{q} \rightarrow y]$ and $y[\mathsf{r} \rightarrow z]$.
`:p rdfs:subPropertyOf :q .`	$x[\mathsf{q} \rightarrow y] :- x[\mathsf{p} \rightarrow y].$

Table 12: Mapping of OWL to the object notation introduced in the thesis; other constructs like `owl:unionOf` or `owl:disjointWith` can be rewritten with the above constructs

Example 24. *We now provide an example transformation from the DL rule of Ex. 11 to an OWL property path axioms. The general principle of the transformation is to transform the tree formed by the rule body into a single path, by rewriting property and class names [39, 67].*

First, we introduce an "is" property for each class name in the rule:

$$x[\mathit{isActuator} \rightarrow x] :- x{:}\mathit{Actuator}.$$
$$x[\mathit{isFluid} \rightarrow x] :- x{:}\mathit{Fluid}.$$
$$x[\mathit{isMechanicalProperty} \rightarrow x] :- x{:}\mathit{MechanicalProperty}.$$
$$x[\mathit{isGeometricProperty} \rightarrow x] :- x{:}\mathit{GeometricProperty}.$$

We then replace all class assertions in the rule with property assertions. From the resulting tree, we can construt the following path from x to z_2:

$$\begin{array}{l}
x[\\
\quad \mathit{isActuator} \rightarrow x,\\
\quad \mathit{actsOnProperty} \rightarrow z_1[\\
\quad\quad \mathit{hasProperty}^{-} \rightarrow y[\\
\quad\quad\quad \mathit{isFluid} \rightarrow y,\\
\quad\quad\quad \mathit{hasProperty} \rightarrow z_2[\\
\quad\quad\quad\quad \mathit{isGeometricProperty} \rightarrow z_2\\
\quad\quad\quad]\\
\quad\quad]\\
\quad]\\
].
\end{array}$$

Note the use of $\mathit{hasProperty}^{-}$ to turn the directed tree into a path. A more readable dot notation can be used for this path:

$$\mathit{isActuator.actsOnProperty.hasProperty}^{-}.$$
$$\mathit{isFluid.hasProperty.isGeometricProperty}$$

Given this path definition, the rule directly translates into OWL as follows:

```
sosa:actsOnProperty owl:propertyChainAxiom (
  ex:isActuator
  sosa:actsOnProperty
  [ owl:inverseOf ssn:hasProperty ]
  ex:isFluid
  ssn:hasProperty
  ex:isGeometricProperty
) .
```

The example we give here is a (simple) particular case: the rule body of ex. 11 has a single branch, when seen as an undirected tree. In the general case, a reflexive "roll-up" property p must be defined for each additional branch $p_1.p_2 \ldots p_n$ such that $x[p \to y]$ unfolds to $x[p_1 \to x_1[p_2 \to \ldots x_{n-1}[p_n \to y]]]$.

B PREFIX & CONTEXT DEFINITIONS

In Ch. 4, we developed a formalization of JSON-LD based on a *context*, that is, a mapping from arbitrary UTF-8 strings to IRIs. All examples in the thesis follow this formalism. Table 13 introduces prefixes for all Web ontology namespaces referenced in examples and Table 14 provides mappings for entity names defined in these namespaces.

NAMESPACE	ONTOLOGY
`rdf:`	RDF Vocabulary
`sosa:`	Sensor, Observation, Sample & Actuation
`schema:`	schema.org
`ssn:`	Semantic Sensor Network
`saref:`	Smart Appliance Reference
`om:`	Ontology of Units of Measure
`bacnet:`	BACowl
`onem2m:`	oneM2M Base Ontology
`ec:`	ecl@ssOWL
`ifc:`	ifcOWL
`brick:`	Brick
`dul:`	Dolce+DnS Ultralite
`owl:`	OWL
`dc:`	Dublin Core
`td:`	Thing Description
`geo:`	WGS84
`bot:`	Building Topology Ontology
`ex:`	*(example namespace, for illustrative purposes)*

Table 13: List of prefixes used in the thesis

TERM	IRI OR CURIE
sensor	`http://example.org/sensor`
Sensor	`sosa:Sensor`
logo	`schema:logo`
Siemens_AG_logo.svg	`http://en.wikipedia.org/...`

TERM	IRI OR CURIE
Sensor	sosa:Sensor
Actuator	sosa:Actuator
Sampler	sosa:Sampler
Platform	sosa:Platform
FeatureOfInterest	sosa:FeatureOfInterest
Procedure	sosa:Procedure
ActuatableProperty	sosa:ActuatableProperty
ObservableProperty	sosa:ObservableProperty
isHostedBy	sosa:isHostedBy
actsOnProperty	sosa:actsOnProperty
observes	sosa:observes
System	ssn:System
Property	ssn:Property
hasSubSystem	ssn:hasSubSystem
implements	ssn:implements
hasProperty	ssn:hasProperty
Thing	td:Thing
InteractionAffordance	td:InteractionAffordance
ReadWritePropertyAffordance	td:ReadWritePropertyAffordance
InvokeActionAffordance	td:InvokeActionAffordance
SubscribeEventAffordance	td:SubscribeEventAffordance
Form	td:Form
hasAffordance	td:hasAffordance
hasForm	td:hasForm
PhysicalObject	dc:PhysicalObject
SpatialThing	geo:SpatialThing
Temperature	om:Temperature
Height	om:Height
WaterTank	ec:C_AKE989002-gen
Heater	ec:C_AAB871002-gen
TemperatureSensor	saref:TemperatureSensor
PneumaticValve	ec:C_AKE773003-gen
FloatSwitch	ec:C_AKE672002-gen
FrequencyConverter	ec:C_AKE176003-gen
Centrifuge	ec:C_AKL741002-gen
Site	bot:Site
Space	bot:Space

TERM	IRI OR CURIE
Radiator	`ex:Radiator`
Wall	`ex:Wall`
hasSpace	`bot:hasSpace`
containsElement	`bot:containsElement`
hasSubElement	`bot:hasSubElement`
hasOrientation	`ex:hasOrientation`
PhysicalBody	`ex:PhysicalBody`
Air	`ex:Air`
Gas	`ex:Gas`
Liquid	`ex:Liquid`
Solid	`ex:Solid`
Fluid	`ex:Fluid`
MechanicalProperty	`ex:MechanicalProperty`
ThermodynamicProperty	`ex:ThermodynamicProperty`
GeometricProperty	`ex:GeometricProperty`
VolumetricFlowRate	`om:Volumetric_flow_rate`
Volume	`om:Volume`
intersects	`schema:geospatiallyIntersects`
within	`schema:geospatiallyWithin`
contains	`schema:geospatiallyContains`
containsZone	`bot:containsZone`
relatesToProperty	`ex:relatesToProperty`
Hall	`ex:Hall`
ConferenceRoom	`ex:ConferenceRoom`
Zone	`bot:Zone`
adjacentElement	`bot:adjacentElement`

Table 14: Global context for the examples in the thesis; other individuals simply use the `tag:` URI scheme (see e.g. Ex. 20)

C SOCIETY & THE WEB OF THINGS

We concluded our thesis by arguing that WoT and formal semantics for WoT can have a positive impact on industrial processes and, more generally, on human societies. Regardless of its nature, these technologies will certainly have a significant impact, which calls for ethical considerations on adopting them. In this mindset and despite being out-of-scope of our thesis, we provide pointers to few works in other academic fields in the form of quotes that relate to cybernetics and various domains of applications of WoT: industry automation, city digitalization, home automation and autonomous driving.

> I do not wish to contribute in any way to selling labor down the river, and I am quite aware that any labor, which is in competition with slave labor, whether the slaves are human or mechanical, must accept the conditions of work of slave labor.

Norbert Wiener, *Father of Cybernetics Norbert Wiener's Letter to UAW President Walter Reuther*, 1949.

> One of the goals of socialism in a libertarian and utopian form is the abolition of the factory by an ecological technology, by creative work, and by cybernetic devices created to satisfy human needs[1].

Murray Bookchin, *Pour un municipalisme libertaire*, 2003.

> [Stafford] Beer acknowledged the difficulties of achieving real-time economic control, but emphasised that a system based on a firm understanding of cybernetic principles could accomplish technical feats deemed impossiblein the developed world, even with Chile's limited technological resources. Once Allende gained a familiarity with the mechanics of Beer's model, he began to reinforce the political aspects of the project and insisted that the system behave in a 'decentralising, worker-participative, and anti-bureaucratic manner'.

Eden Medina, "Designing Freedom, Regulating a Nation: Socialist Cybernetics in Allende's Chile", *Journal of Latin American Studies*, 2006.

1 translation is by myself.

The city is a symbol of outward-looking cosmopolitanism – a potent answer to the homogeneity and insularity of the nation state. Today it is the only place where the idea of exerting meaningful democratic control over one's life, however trivial the problem, is still viable.

From transport to food delivery, from accommodation to energy consumption, the city also figures prominently in how digital technologies penetrate our life.

Evgeny Morozov, "There is a leftwing way to challenge big tech for our data. Here it is," *The Guardian*, August 19th 2018.

This must be what it's like to be in a documentary or in a reality TV show. The cameras eventually move to the periphery of your vision and then disappear altogether. If homes become sentient, and it becomes the norm that activity in them is captured, measured, and used to profile us, all of the anxiety you currently feel about being tracked online is going to move into your living room.

Kashmir Hill & Surya Mattu, "The House that Spied on Me," *Gizmodo*, July 2nd 2018.

[George Dyson:] "There's this old law called Ashby's law that says a control system has to be as complex as the system it's controlling, and we're running into that at full speed now, with this huge push to build self-driving cars where the software has to have a complete model of everything, and almost by definition we're not going to understand it."

Andrew Smith, "Franken-algorithms: the deadly consequences of unpredictable code," *The Guardian*, August 30th 2018.