Characterizing Web of Things Interactions with Existential Reasoning

Victor Charpenay [a,b,*], Sebastian Käbisch [a] and Harald Kosch [b]
[a] *Corporate Technology, Siemens AG, Germany*
E-mails: victor.charpenay@siemens.com, sebastian.kaebisch@siemens.com
[b] *Faculty of Computer Science and Mathematics, University of Passau, Germany*
E-mails: victor.charpenay@siemens.com, harald.kosch@uni-passau.de

Abstract. The Web of Things (WoT) is a collection of interlinked Web resources exposed by autonomous sensors and actuators that interact to perform complex automation tasks. This paper presents a method to characterize interactions on the Web of Things in terms of relations between these devices and the physical world entities that compose their environment. In particular, based on the recent standardization by the W3C of the Thing Description (TD) model and its alignment with other Web ontologies, devices expose logical assertions on themselves that form a knowledge graph from which a 'graph of interactions' can be derived.

The reasoning task of interest in WoT is query answering over ontologies that feature existential restrictions on the 'things' WoT devices observe or act upon. Because no complete algorithm exists for this task, we present a tractable skolemization algorithm for the ELP fragment of Description Logics (DLs), at the intersection of $\mathcal{EL}^{++}$ and Datalog. We tested our approach on two use cases in different industry domains: Building Automation (BA) and Industrial Control Systems (ICS).

Keywords: Existential Reasoning, Description Logic, ELP, Web of Things, Thing Description, Semantic Discovery

1. Introduction

One of the promises of the Web of Things (WoT) is to bring more autonomy in automation systems by the automatic mash-up of Web agents [1]. These agents, mostly embedded devices, are capable of sensing and/or acting on specific aspects of the physical world and provide a Web interface to them. Application mash-ups in WoT can essentially be viewed as multi-agent systems (MAS), as are other ubiquitous and pervasive systems [2]. As such, a WoT system is first described by the *interactions* that take place between WoT agents, acting either as clients, servers or both ('servient' is the generic term to designate WoT agents [3]). It is expected that typical WoT systems involve a large and heterogeneous fleet of servients. For instance, Building Automation (BA) systems, which are one of the identified application areas of WoT, typically involve thousands of control devices per building. This necessarily leads to complex interaction patterns between WoT servients.

One approach to address this complexity is to characterize interactions in terms of knowledge. First, one can define a 'graph of interactions' for a given WoT system such that its vertices are servients and there is an edge between vertices whenever the associated servients interact in the system. Then, one can formulate the hypothesis that there is a similarity between this graph of interactions and some knowledge graph describing the system and its environment. Early on, knowledge graphs and, more precisely, Semantic Web technologies had been identified as relevant in the broader field of the Internet of Things (IoT) [4]. In WoT, this view resulted in the standardization by the W3C of the Thing Description (TD) model, a simple RDF model to semantically describe 'things' and their interaction affordances via servients [5, 6].

*Corresponding author. E-mail: victor.charpenay@siemens.com.

1.1. Identification on the Web of Things

The main challenge of a knowledge-base approach to characterizing WoT interactions is the construction of the knowledge graph to compare with. More precisely, it is not obvious to *identify* the relevant physical world entities, like walls, pieces of furniture and other objects that might influence BA systems. Most approaches from the IoT to this problem consist in providing methods for digitally "tagging" physical world objects with electronic chips. The relevant technologies in that respect are e.g. Near-Field Communication (NFC), Zigbee or Bluetooth Low-Energy (BLE). These technologies provide short-range communication protocols to expose information in a machine-readable format. One can also think of the use of bar codes to reference consumer goods as the ancestor of the IoT for it is also a means to digitally identify these goods.

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, 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]. For physical world entities that are not directly exposed on the Web, URI schemes like `tag:` [7], `urn:` [8] or `ni:` [9] can be used. Interestingly, a URI scheme for bar codes is currently being specificied in the perspective of being integrated into WoT [10].

However, the general approach of tagging comes with high deployment efforts and maintainance costs. Moreover, a collection of IRIs is arguably not enough to make a "web" of Things. Tagging cannot provide links between IRIs, expressing e.g. a containment relation between a radiator and a room or adjacencies between rooms. In contrast, well maintained Web ontologies like ifcOWL [11], Brick [12] or BOT [13] provide general knowledge about buildings and other application domains for WoT, which can be used to infer statements about 'things' from TD documents giving e.g. the location of a sensor in a building.

In this paper, we develop a framework based on *existential reasoning* to characterize WoT interactions. That is, these Web ontologies include axioms that only refer to the existence of physical world entities without strictly identifying them. For instance, the axiom 'every room in a building has walls' is an existential restrictions on all rooms. There exist numerous works on this kind of automated reasoning, as our review of the state-or-the-art shows, especially around Web ontologies (Sec. 2). We use and extend some of the techniques found in the literature to infer the existence of physical world objects from knowledge graphs describing WoT systems. In particular, we present meaningful relations one can infer to find possible interactions between WoT agents (Sec. 3). We tested this approach on two different use cases, in the BA domain with a dataset provided by Intel labs (Sec. 4.1) and in the domain of industry automation, on a water treatment plant model (Sec. 4.2).

Throughout this paper, we will reference various Web ontologies, listed in Table 1. All terms appearing in examples are defined by one of these models, if not specified otherwise. For the sake of conciseness, we omit most prefixes.

Table 1
Web ontologies mentioned in the present paper

Name	Prefix	Ref.
Sensor, Observation, Sample & Actuator	`sosa`	[14]
Semantic Sensor Network	`ssn`	[14]
Units of Measure	`om`	[15]
Building Topology	`bot`	[13]
eCl@ss (Products and Services)	`ec`	[16]
schema.org	`schema`	[17]

2. Related Work: Identification by Reasoning

Broadly speaking, reasoning is the process of drawing conclusions from certain knowledge [18]. 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 both cases, it is necessary that observations are correctly interpreted in order for actuators to be controlled properly.

In computational logic, every reasoning task can be reduced to the problem of *satisfiability* of a proposition (a formula) f by a set of propositions (a knowledge base) $\mathcal{K}$, denoted $\mathcal{K} \models f$. 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. We can reason-

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

ably assume that available TDs in a WoT system only cover a limited extent of all the possible physical world entities that can be perceived. Therefore, the reasoning task of interest in WoT is that of existential reasoning: assuming the existence of some entity not described by a TD, does $\mathcal{K} \models f$ still hold?

In the following, we briefly review RDF graph canonicalization as an existential reasoning technique and then consider more in detail the case of existential reasoning with Web ontologies in terms of satisfiability, query answering and query abduction.

2.1. RDF Graph Canonicalization

The RDF data model includes the notion of *blank node*. According to the *RDF 1.1 Semantics* specification document [19]:

> 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.

An algorithm based on node coloring has recently been developed for canonicalization [20, 21]. We briefly explain its principle with an example.

Example 1. *Consider the following description of two radiators in room 31.638 of the Siemens Legoland campus, expressed in terms of BOT classes and properties (Turtle syntax):*

```
<tag:legoland> a bot:Site ;
  bot:hasSpace <tag:31.638> .
<tag:31.638> a bot:Space ;
  bot:containsElement _:r1, _:r2 .
_:r1 a ex:Radiator .
_:r2 a ex:Radiator .
```

This example includes the blank nodes `_:r1`, `_:r2`. 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.

To be able to distinguish between the two radiators, we could .e.g. precise on which wall they are mounted.

Example 2. *Let us update Ex. 1 with South and East walls.*

```
<tag:legoland> a bot:Site ;
  bot:hasSpace <tag:31.638> .
<tag:31.638> a bot:Space ;
  bot:containsElement _:southWall,
                      _:eastWall .
_:southWall a ex:Wall ;
  ex:hasOrientation <tag:south> ;
  bot:hasSubElement _:r1 .
_:eastWall a ex:Wall ;
  ex:hasOrientation <tag:east> ;
  bot:hasSubElement _:r1 .
_:r1 a ex:Radiator .
_:r2 a ex: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. Any additional property on 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

```
<tag:31.638>
  bot:containsElement _:southWall .
_:southWall
  bot:hasSubElement _:r1 .
```

then, transitively, it is also true that

```
<tag:31.638>
  bot:containsElement _:r1 .
```

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.

2.2. Reasoning with Web ontologies

2.2.1. Description Logics

The theoretical foundations the Web Ontology Language (OWL) are Description Logics (DLs), a family of logic formalisms developed in parallel to classical First-Order Logic (FOL). In the following, we provide formal definitions for DL knowledge bases and review 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], we keep using OWL terms throughout this paper.

In all our definitions, we define N^C, N^P and N^I as respectively the set of class names, property names and individual names (all elements of these sets being IRIs). We also define a set of variables, denoted V. We can now formally define a DL knowledge base.

Definition 1. *[22] Let $t, t' \in V \cup \mathsf{N}^I$, let $p \in \mathsf{N}^P$ and $C \in \mathsf{N}^C \cup \{\top, \bot\}$. A DL expression is a formula f of the form*

$$C(t) \mid \exists p.C(t) \mid p(t,t') \mid f' \wedge f''$$

where f', f'' are themselves DL expressions. A DL knowledge base is a set of rules $B \rightarrow H$ where H (the rule head) is either of the form $C(t)$, $\exists p.C(t)$ or $p(t,t')$ and B (the rule body) is a DL expression, such that

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

The top class $\top$ (resp. bottom class $\bot$) is a special class defined as the super-class (resp. sub-class) of every class in N^C. Formally, we have the rules $C(x) \rightarrow \top(x)$ and $C(x) \rightarrow \bot(x)$ for all $C \in \mathsf{N}^C$ 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 we call a CBox.

Example 3. *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.*

$$\texttt{Space}(x) \rightarrow \texttt{Air}(x).$$

$$\texttt{Air}(x) \rightarrow \texttt{PhysicalBody}(x).$$

$$\texttt{PhysicalBody}(x) \rightarrow \exists \texttt{hasProperty}.\texttt{Temperature}(x).$$

The existential quantifier ($\exists$) expresses an *existential restriction* on the class `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 $\mathcal{EL}^{++}$ [23, 24]. Definition 1 subsumes both logics in terms of expressiveness. There exist more expressive DLs featuring e.g. class complements ($\neg C$) and inverse properties (p^-) but the DL fragment we consider 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 [25]. In contrast, the much more expressive FOL is undecidable, which is one of the reasons why DLs were chosen 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. We do not provide a formal definition of these restrictions here, the reader can safely assume they apply to all examples shown in this paper. This rule-based notation is inspired from earlier observations that all DL axioms can be expressed as rules of a certain form [26, 27]. We refer to Krötsch's *Description Logic Rules* book for an exhaustive definition of DL rules [26].

2.2.2. Semantics of Description Logics

In order to decide whether $\mathcal{K} \models \alpha$ for some $\mathcal{K}, \alpha$, we must define the *semantics* of DL expressions. We do so 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 what it perceives from it to arbitrary symbols, i.e. by interpreting it. Formally. an interpretation $\mathcal{I}$ is composed of some (arbitrary) domain of interpretation, denoted $\Delta^{\mathcal{I}}$, and some interpretation function denoted $\cdot^{\mathcal{I}}$ that maps class, property and individual names to $\Delta^{\mathcal{I}}$ [25]. More precisely, for a class name $C \in \mathsf{N}^C$, we have $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$, for a property name $p \in \mathsf{N}^P$, we have $p^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ and for an individual name $a \in \mathsf{N}^I$, we have $a^{\mathcal{I}} \in \Delta^{\mathcal{I}}$.

[2] In particular, classes and properties are called concepts and roles in the DL literature

Definition 2. *[22] Let f be a DL expression as per Def. 1. 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 $\sigma : V \cup \mathsf{N}^I \mapsto \Delta^{\mathcal{I}}$, such that $\sigma(a) = a^{\mathcal{I}}$ for all $a \in \mathsf{N}^I$ 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 $B \rightarrow H$ if $\mathcal{I}$ satisfies H or $\mathcal{I}$ fails to satisfy 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 $\mathcal{I} \models \mathcal{K}$.

Example 4. *In our previous example, one can observe that adding the following rule to $\mathcal{K}_{ex}$, the knowledge base is still satisfiable:*

$$\texttt{Space}(x) \rightarrow \exists \texttt{hasProperty}.\texttt{Temperature}(x).$$

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

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 [28]. 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) [29]. 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 allow existential restrictions, though. This limitation motivated the definition of ELP, which combines DLP with $\mathcal{EL}^{++}$ axioms, as well as some other Datalog constructs [22]. It is similar, yet more expressive than another logic allowing for existential reasoning over Datalog, called Datalog$^{\pm}$ [30].

Figure 1 shows the mutual inclusion of all these formalisms in terms of epxressivity (taken to the most part from the foundation paper on Datalog$^{\pm}$ by Calí et al. [30]). 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. In Sec. 3.1, we will discuss what formalism is suitable for existential reasoning in WoT use cases.

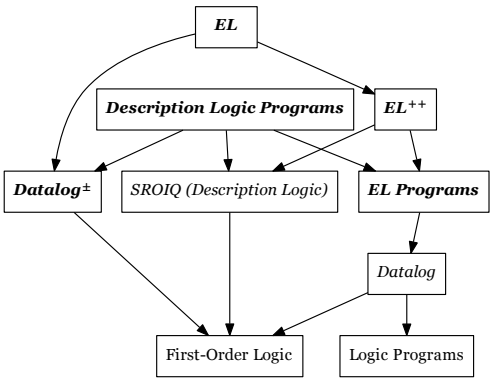


Fig. 1. Partial order in terms of expressivity between logic formalisms related to DLs with existential restrictions (tractable fragments in bold font, decidable fragments in italic for the problem of satisfiability)

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

2.3. Query Answering and Abduction

2.3.1. Conjunctive Query Answering

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

Definition 3. *[31] A DL conjunctive query (CQ) $\mathcal{Q}$ is a DL expression without existential restrictions, that is, a conjunction of expressions of the form $C(x)$ and $p(x,y)$ with $x, y \in V$, $C \in \mathsf{N}^C$ and $p \in \mathsf{N}^P$. A knowledge base $\mathcal{K}$ entails $\mathcal{Q}$ if for all model $\mathcal{I}$ of $\mathcal{K}$, $\mathcal{I}$ also satisfies all expressions in $\mathcal{Q}$. The same notation is used for satisfiability and entailment, we write $\mathcal{K} \models \mathcal{Q}$.*

Example 5. *For example, the following conjunctive query is indeed entailed by $\mathcal{K}_{ex}$:*

$$\texttt{Space}(x) \wedge \texttt{Temperature}(y) \wedge \texttt{hasProperty}(x, y).$$

However, such a result is of little interest if the knowledge base does not include any ABox assertion. If we add the assertion $f = \texttt{Space}(\texttt{31.638})$ to $\mathcal{K}_{ex}$, then $\mathcal{K}_{ex} \cup \{f\}$ entails the following CQ:

$$\texttt{hasProperty}(\texttt{31.638}, y) \wedge \texttt{Temperature}(y).$$

Example 5 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, we denote $\mathsf{N}_{\mathcal{K}}^{I}$ the set of individual names in $\mathcal{K}$. We also denote var($\mathcal{Q}$) the set of variables in $\mathcal{Q}$.

Definition 4. *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 $x \in V$ in $\mathcal{Q}$ with an individual name $a \in \mathsf{N}_{\mathcal{K}}^{I}$, 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 $var(\mathcal{Q}'') \subseteq var(\mathcal{Q}')$.*

It is easy to see that substitutions closely relate to the function σ defined in Def. 2. Indeed, given an model $\mathcal{I}$ of $\mathcal{K}$, a substitution exists for some $a \in \mathsf{N}_{\mathcal{K}}^{I}$ if and only if $\sigma(x) = a^{\mathcal{I}}$. 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 conjunctive query* (BCQ). In practice, it is common to answer a CQ by testing the entailment of 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: even if the temperature property of room 31.638 is also asserted in $\mathcal{K}_{\text{ex}}$, we argued in introduction that most physical world objects have no explicit TD. *A fortiori*, their properties will also not be asserted. For instance, in the CQ of Ex. 5, 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: expressions to be satisfied can be considered as simple CQ, without conjunctions. In contrast, CQs are considered complex when they include at least one conjunction with shared variables between formulas [32]. It is comparable to queries with joins in relational databases. In fact, DL query answering shares many aspects with relation algebra. The DL subset that gained most attention with respect to query answering is called DL-Lite [33]. 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. [34]). 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. For a DL fragment that excludes class disjunctions and complements, as well as transitive and functional properties [32], DL query answering is tractable for the data complexity. This fragment, which has much in common with Datalog$^{\pm}$, is subsumed by Def. 1. 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 $\mathcal{EL}^{++}$ (which features transitive properties) is also tractable [31]. Later in this paper, we further extend this result and present a tractable query answering algorithm for knowledge bases as defined in Def. 1 (Sec. 3.3).

To conclude this section on DL query answering, we define the closely related notion of *explanation*, which is 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.

Definition 5. *[25] 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 sometimes 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.

2.3.2. Abductive Reasoning

Abduction can be described in general terms as the procedure of finding missing assertions (or rules) 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" assertion is asserted in $\mathcal{K}$ [35]. However, there are infinitely many such assertions 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 [36]. Early formalisms based on logic programming also involve *integrity constraints* allowing or not certain assertions [35]. 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 6. *[36] 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:*

- $\mathcal{K} \cup \mathcal{K}' \models \mathcal{Q}$
- $\mathcal{K} \cup \mathcal{K}' \not\models \bot$
- $\mathcal{KB}' \not\models \mathcal{Q}$

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 $\mathsf{N}^I_{\mathcal{K}}$ and a finite set of "fresh" IRIs (in order to obtain BCQs) and then applying Def. 6.

There exist various works addressing abduction on different DL fragments, with focus on tractability [36–38]. All of these works reduce the solution space to rules with an empty head (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 our knowledge, there is no algorithm covering Def. 1.

As we saw in our review for query answering, there is no known implementation to correctly process existential restrictions. One possible strategy consists in formulating 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 [39]. This experiment suggests that this approach performs well on large ABoxes.

In an earlier publication, we made an attempt to formalize and implement reasoning with TDs using ALP [40]. The underlying DL was $\mathcal{EL}^{++}$. However, as we will show in Sec. 3.1, $\mathcal{EL}^{++}$ proved impractical in the use cases we consider here as in the absence of equality relations between anonymous individuals, we have an explosion of the solution space as the ABox of the knowledge base grows. Regardless of any optimization known for ALP, computing all answers is then highly expensive. Yet, there is no state-of-the-art algorithm for DLs more expressive than $\mathcal{EL}^{++}$. 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.

In the following, we present the problem of WoT semantic discovery, formulated as an existential reasoning problem. We chose to formulate our problem only in terms of query answering; possible abduction-based optimizations are left as future work.

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 descriptions of physical world objects and their properties (possibly implicit). Semantic discovery accepts as input a set of TDs and domain-specific knowledge provided by Web ontologies. We have already mentioned several ontologies without considering, however, their completeness in terms of axiomatization to solve problems like semantic discovery. We first review axioms potentially missing in these ontologies, before moving on to a formalization of semantic discovery and then to implementation considerations.

3.1. Expressiveness of a Web of Things Knowledge Base

3.1.1. Expressing Existence and Equality Between Individuals

As mentioned in introduction, the focus of this paper is identification by reasoning as opposed to iden-

tification by physical tagging. Our basic assumption is that Web ontologies provide generic axioms that can be exploited to identify anonymous entities from assertions found in TDs. The axioms of interest in our context are those expressing the existence of some anonymous individuals and equality between them. These OWL axioms translate into rules in our present formalism. Formally, an axiom contributes to expressing existence or equality if all model $\mathcal{I}$ of some knowledge base $\mathcal{K}$ has some characteristic only in the present of the corresponding rule(s). In the case of existence, it means that for all model $\mathcal{I}$ of $\mathcal{K}$, there is some $\delta \in \Delta^{\mathcal{I}}$, such that (1) $\sigma(x) = \delta$ for some $x \in V$ and (2) there is no named individual $a \in \mathsf{N}^I$, such that $a^{\mathcal{I}}_{\mathcal{K}} = \delta$, i.e. δ is anonymous. In the case of equality, it means that there is $\langle \delta_1, \delta_2 \rangle \in p_1^{\mathcal{I}}, \langle \delta_3, \delta_4 \rangle \in p_2^{\mathcal{I}}$ for some $p_1, p_2 \in \mathsf{N}^P$ such that there is the equality relation $\delta_1 = \delta_3$ or $\delta_2 = \delta_4$ between anonymous individuals. First, we briefly review various OWL features as to whether they express existence, equality or none of these aspects in order to decide on what DL fragment is most relevant in WoT. Then, we give examples of such axioms with classes from the WoT ontology cloud.

In Table 2, we give a list of the DL features that have a direct correspondance to OWL. All features can be simply expressed with DL rules if we add class complements ($\neg C$) to Def. 1. First, it is easy to see that universal restrictions do not contribute to existential reasoning, either for existence or equality. Indeed, for some class C, an interpretation $\mathcal{I}$ can satisfy restrictions on $C^{\mathcal{I}}$ regardless of whether $C^{\mathcal{I}}$ is empty or not. The observation also holds for class complements and maximum cardinality constraints. On the other hand, minimum cardinality constraints are relevant since they subsume existential restrictions, the only dedicated construct to express existence. Finally, one can note that class complements may indirectly express equality, by negation.

With respect to properties, DL features only contribute to equality. However, these features are key in WoT since they allow for the construction of relations between individuals that correspond to systems via anonymous individuals that represent physical world entities. This pattern is then exploited to infer possible interactions between these systems.

Functional properties, by definition, are used to identify entities, like primary keys in a database. Property chains are the main construct to infer more knowledge about anonymous individuals, beside their mere existence. Therefore, they are important for equality. Transitivity is a special case of property chain. Reflexive and inverse properties do not directly contribute to expressing existence or equality but they do simplify the axiomatization of Web ontologies. We will discuss their usefulness later in this section. The same holds for symmetric properties, a special case of inverse properties.

Table 2

DL features for properties (top) and classes (bottom) expressing the existence ($\exists$) and equality ($=$) between anonymous individuals; the number of occurrences for each feature in the ontologies from Table 1 is also given

DL feature	$\exists$	$=$	Occurrences
Existential restriction	yes	no	14
Universal restriction	no	no	215
Min. cardinality constraint	yes	no	0
Max. cardinality constraint	no	no	0
Class complement	no	yes	0
Transitive property	no	yes	1
Functional property	no	yes	14
Inverse property	no	no	35
Symmetric property	no	no	2
Reflexive property	no	no	0
Property chain	no	yes	6

3.1.2. *Examples for Physical Bodies and their Properties*

Table 2 also shows occurrences of the different features in the ontologies of Table 1. 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 (i.e. its class, property and individual names). We present examples of rules in the following, on two kinds of physical world entities: properties of features of interest (like a temperature) and objects related to the features of interest asserted in TDs (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. On Fig. 2, we give an example of classification for physical bodies that extends OM, the ontology of units of measure. Physical bodies are distinguished according to their phase (liquid, solid or gas). Every physical body has a `om:Volume` and a `om:Temperature` but only fluids (liquid, gas) have a `om:Volumetric_flow_rate`. All examples of rules in Sec. 2 come from these axioms.

Physical quantities are further categorized according to the domain of physics that defines them. For

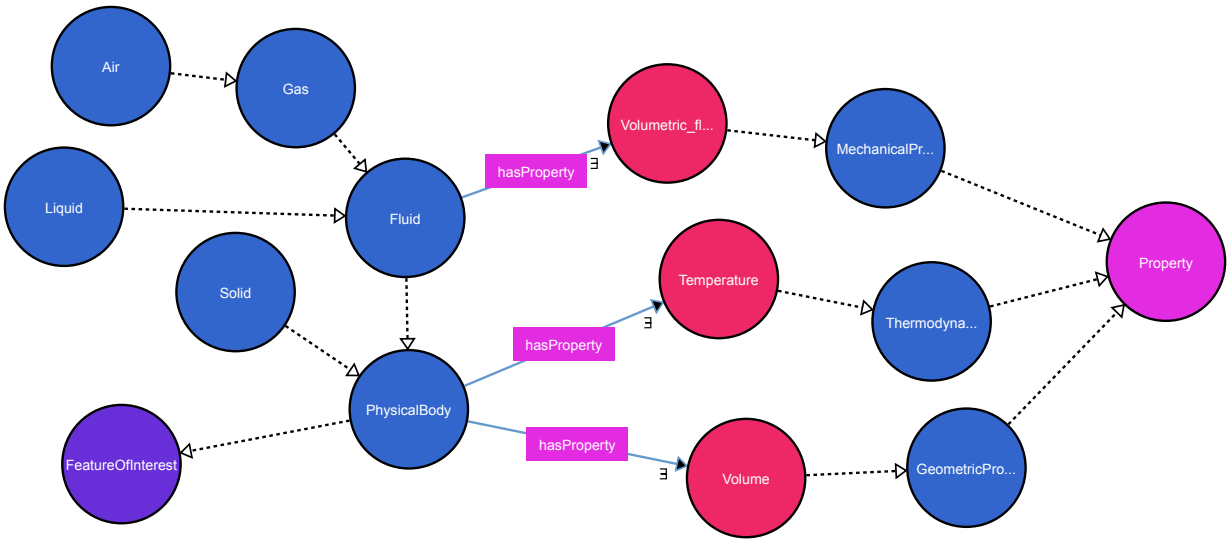


Fig. 2. Knowledge base for physical bodies and their properties (VOWL syntax [41])

instance, volume is relevant in geometry, property in thermodynamic and flow rate in mechanics. These categories help expressing equality relations between properties of distinct objects in an indirect manner.

Example 6. *The following rule states for instance that two intersecting fluids share the same thermodynamic properties:*

$$\texttt{Fluid}(x_1) \wedge \texttt{Fluid}(x_2) \wedge \texttt{intersects}(x_1, x_2) \wedge \texttt{hasProperty}(x_2, y) \rightarrow \texttt{hasProperty}(x_1, y).$$

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

$$\texttt{Actuator}(x) \wedge \texttt{Fluid}(y) \wedge \texttt{actsOnProperty}(x, z_1) \wedge \texttt{hasProperty}(y, z_1) \wedge \texttt{hasProperty}(y, z_2) \wedge \texttt{MechanicalProperty}(z_1) \wedge \texttt{GeometricProperty}(z_2) \rightarrow \texttt{actsOnProperty}(x_1, y).$$

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

Example 8. *An example can be found in BOT, where* `bot:containsZone` *property is declared as transitive:*

$$\texttt{containsZone}(x, y) \wedge \texttt{containsZone}(y, z) \rightarrow \texttt{containsZone}(y, z).$$

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

Example 9. *For instance, the following rule states that a solid and a fluid that are both within some physical body necessarily intersect:*

$$\texttt{Solid}(x) \wedge \texttt{PhysicalBody}(y) \wedge \texttt{Fluid}(z) \wedge \texttt{within}(x, y) \wedge \texttt{contains}(y, z) \rightarrow \texttt{intersects}(x, z).$$

Geo-spatial relations in Ex. 9 are provided by schema.org (`schema:geospatiallyWithin` and `schema:geospatiallyContains`).

From Table 2, the feature we used to express these DL rules are property chains and (indirectly) in-

verse and reflexive properties [3]. 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. Since inverse properties do not directly contribute to existential reasoning, we explicitly left them out in our formalization of DL knowledge bases (recall Def. 1). We also discarded functional properties and class complements, although these features may impact equalities among anonymous properties. Experimentally, we have never been confronted with the case where they were unavoidable (contrary to property chains, which proved crucial). It is unclear, however, if it holds in the general case.

Next, we formalize a WoT-specific problem based on existential reasoning on the rules we have just presented. This formalization holds regardless of the DL features we allow or not in a knowledge base. The choice we have made in that respect is only relevant when considering practical implementation (Sec. 3.3).

3.2. Problem Statement

Like all multi-agent systems, WoT systems are primarily defined by the interactions that take place between servients. The general assumption of this paper is that the graph of interactions $G = \langle V, E \rangle$, such that V is a set of servients and E a set of unordered pairs $\{x, y\}$ whenever servient x interacts with servient y, is directly related to the knowledge base $\mathcal{K}$ describing the system and its environment. More precisely, we assume that every servient interaction relates to a specific CQ $\mathcal{Q}$ involving both servients as instances of `ssn:System`, such that $\mathcal{K} \models \mathcal{Q}$. In other words, every edge in G is labeled with some CQ.

Example 10. *Consider a HVAC system with two servients `s1`, `s2` respectively controlling a radiator and a temperature sensor, both located in room 31.638. The two devices are synchronized to keep the temperature constant. We then have* $G = \langle \{s1, s2\}, \{e = \{s1, s2\}\} \rangle$ *and e is labeled with the following query:*

$$\texttt{observes}(s1, x) \wedge \texttt{actsOnProperty}(s2, x).$$

[3] we refer to Ch. 8 of *Description Logic Rules* for an explanation on how to transform DL rules to so-called "qualified" property chains.

In this paper, unlike traditional MAS studies, we do not consider the (possibly contradictory) goals of WoT servients, nor do we consider the events to which they react. Instead, we only aim at preserving the logical consistency of the system by guaranteeing that every interaction can be *explained* in terms of conjunctive querying. As a consequence, edges of our graph of interactions are undirected: we do not consider whether the client initiates the request or if it receives an asynchronous notification from the server. 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 (in which case they are called *peer-to-peer* interactions).

Given our formalisms for graphs of interactions, several problems can be defined. For instance, given a graph of interactions G and a knowledge graph $\mathcal{K}$, one problem is that of labeling G with minimal CQs. Another problem of interest is to find the biggest graph G for a given knowledge base $\mathcal{K}$ and a query $\mathcal{Q}$. We refer to the latter as *semantic discovery* and formalize it in the following.

Our basic assumption is that every WoT servient $\mathtt{s}_i$ exposes in the form of a TD an ABox $\mathcal{A}_i$ (a set of assertions) with respect to a shared CBox $\mathcal{C}$ (a set of rules). We denote $\mathcal{A}$ the ABox $\bigcup_i \mathcal{A}_i$ and $\mathcal{K}$ the knowledge base defined as $\mathcal{A} \cup \mathcal{C}$.

Definition 7. *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* $\mathcal{Q}'$ *of* $\mathcal{Q}$ *that includes* $a \in \mathsf{N}^I_{\mathcal{A}_i}$ *and* $b \in \mathsf{N}^I_{\mathcal{A}_j}$ *distinct such that* $\mathcal{K} \models \texttt{System}(a) \wedge \texttt{System}(b)$.

The biggest graph of interactions for $\mathcal{K}$ *then has an edge* $\{a, b\}$ *labeled with* $\mathcal{Q}'$.

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

Example 11. *Let us define some minimal TD for a radiator (*$\mathcal{A}_1$*):*

$$\texttt{Radiator}(s1).$$
$$\texttt{within}(s1, 31.638).$$
$$\texttt{Space}(31.638).$$

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

$$\texttt{TemperatureSensor}(s2).$$

$\mathit{within}(\mathit{s2}, \mathit{31.638})$.

$\mathit{Space}(\mathit{31.638})$.

and a CBox similar to what we presented in Sec. 3.1 ($\mathcal{C}$):

$\mathit{Radiator}(x)$
$\quad \rightarrow \exists \mathit{actsOnProperty}.\mathit{Temperature}(x)$.

$\mathit{TemperatureSensor}(x)$
$\quad \rightarrow \exists \mathit{observes}.\mathit{Temperature}(x)$.

$\mathit{Space}(x)$
$\quad \rightarrow \exists \mathit{hasProperty}.\mathit{Temperature}(x)$.

$\mathit{within}(x,y) \wedge \mathit{hasProperty}(x,z) \wedge$
$\mathit{Temperature}(z) \wedge$
$\exists \mathit{actsOnProperty}.\mathit{Temperature}(y)$
$\quad \rightarrow \mathit{actsOnProperty}(x,z)$.

$\mathit{within}(x,y) \wedge \mathit{hasProperty}(x,z) \wedge$
$\mathit{Temperature}(z) \wedge$
$\exists \mathit{observes}.\mathit{Temperature}(y)$
$\quad \rightarrow \mathit{observes}(x,z)$.

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

$\mathit{observes}(x,z) \wedge \mathit{actsOnProperty}(y,z)$.

The output of semantic discovery on $\mathcal{K}$ for $\mathcal{Q}$ is the graph of interactions given in Ex. 10.

In practice, for a broader graph of interactions taking into account sensor/sensor and actuator/actuator interactions, we can introduce the property `relatesToProperty`, as follows:

$\mathtt{observes}(x,y)$
$\quad \rightarrow \mathtt{relatesToProperty}(x,y)$.

$\mathtt{actsOnProperty}(x,y)$
$\quad \rightarrow \mathtt{relatesToProperty}(x,y)$.

and then run semantic discovery on the following query:

$\mathtt{relatesToProperty}(x,z) \wedge$
$\mathtt{relatesToProperty}(y,z)$.

This is the query we used in our experiments, presented later (Sec. 4).

When an interaction indeed takes place in a system, it can be stated by a `ssn:hasSubSystem` relation between a composite system and its sub-systems, either interacting in a mediated or peer-to-peer fashion (where the composite system is virtual). In the latter case, we assume that TDs are managed by the servients themselves, as opposed to being stored in a central RDF store. As a consequence, these TDs must be updated with the results of semantic discovery, in order for all servients involved in a new system to have sufficient knowledge about it. That is, each servient involved in a potential interaction should be able to entail the associated CQ from its own ABox. This relates to the notion of query explanation we defined earlier (Def. 5).

Definition 8. *Let $\mathcal{E}$ be an explanation for a CQ $\mathcal{Q}$ on $\mathcal{K}$. The update ABox for $\mathcal{Q}$ in a peer-to-peer system is the ABox $\mathcal{A}' = \bigcup_i \mathcal{A}'_i$, such that for all $\mathcal{A}_i$ where $\mathsf{N}^I_{\mathcal{A}_i} \cap \mathsf{N}^I_{\mathcal{E}} \neq \emptyset$, $\mathcal{A}'_i = \mathcal{E} \setminus (\mathcal{A}_i \cup \mathcal{C})$.*

Given Def. 8, 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 12. *The update ABox for Ex. 11 is the union of the following ABox for the radiator ($\mathcal{A}'_1$):*

$\mathit{Radiator}(\mathit{s1})$.

$\mathit{within}(\mathit{s1}, \mathit{31.638})$.

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

$\mathit{TemperatureSensor}(\mathit{s2})$.

$\mathit{within}(\mathit{s2}, \mathit{31.638})$.

In this particular case, $\mathcal{A}'_1 = \mathcal{A}_2$ and $\mathcal{A}'_2 = \mathcal{A}_1$ but in the general case, the update ABox for a given servient is inversely proportional to its prior level of knowledge.

3.3. A Tractable Approach

In Ex. 11, 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 our review of the state-of-the-art in DL reasoning (Sec. 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. We define the SKOLEMIZE procedure in the following (Alg. 1) and prove the equivalence in terms of query answering between a knowledge base and its skolemization (Th. 1).

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.

function SKOLEMIZE($\mathcal{K}$, $\mathsf{S} \subseteq \mathsf{N}^I_{\mathcal{K}}$, n)
  Let $\mathcal{K}' := \mathcal{K}$, $\mathsf{S}' := \emptyset$
  for all $a \in \mathsf{S}, p \in \mathsf{N}^P_{\mathcal{K}}, C \in \mathsf{N}^C_{\mathcal{K}}$ **do**
    if $\mathcal{K} \models \exists p.C(a)$ **then**
      Let b be a fresh individual
      $\mathcal{K}' := \mathcal{K}' \cup \{p(a,b), C(b)\}$
      $\mathsf{S}' := \mathsf{S}' \cup \{b\}$
    end if
  end for
  if $n = 1$ **then**
    for all f in $\mathcal{K}$ of the form $\exists p.C(t)$ **do**
      Let x be a fresh variable
      $f' := p(t,x) \wedge C(x)$
      Replace all occ. of f in $\mathcal{K}'$ with f'
    end for
    return $\mathcal{K}'$
  else
    return SKOLEMIZE($\mathcal{K}'$, S', $n-1$)
  end if
end function

Theorem 1. *Let $\mathcal{K}$ be a DL knowledge base, i.e. a set of rules of the form $B \rightarrow H$. 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 $\mathcal{K}$, $\mathsf{N}^I_{\mathcal{K}}$ 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 $\langle a^{\mathcal{I}}, \delta_1\rangle, \ldots \langle \delta_{k-1}, \delta_k\rangle$ is at most of length n, for some $a \in \mathsf{N}^I_{\mathcal{K}}$ and $\delta_1, \ldots, \delta_k$ anonymous. Proof: for all rule $B \rightarrow H$ in $\mathcal{K}$, s.t. $\mathcal{I}$ satisfies both H and B, all property chains in B are of the form $p_1(t_1,t_2), \ldots, p_{n-1}(t_n,t_n)$ (as per Def. 1). 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 [25]: (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. 2 for $\mathcal{I}$, then it also satisfies all formulas of $\mathcal{K}$ with no existential restriction. If a formula includes a restriction $\exists p.C(t_i)$ in $\mathcal{K}$, then there is an equivalent formula $p(t_i,x) \wedge C(x)$ 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 $\mathsf{N}^C_{\mathcal{K}} \cup \mathsf{N}^P_{\mathcal{K}} \cup \mathsf{N}^I_{\mathcal{K}}$. 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) [34]. If $\sigma(x)$ is anonymous for some variable x in $\mathcal{K}$, then there is a path $\langle a^{\mathcal{I}}, \delta_1\rangle, \ldots, \langle \delta_{k-1}, \delta_k\rangle$ 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 $p_1, \ldots, p_k$, we have $\langle a^{\mathcal{I}}, \delta_1\rangle \in p_1^{\mathcal{I}}$, $\ldots$, $\langle \delta_{k-1}^{\mathcal{I}}, \delta_k\rangle \in p_k^{\mathcal{I}}$ and for some $C_1, \ldots, C_k$, we have $\delta_1 \in C_1^{\mathcal{I}}, \ldots, \delta_k \in C_k^{\mathcal{I}}$, in order for this path to be unique (otherwise, it would be possible to define μ). It follows that $\exists p_1.C_1(a)$ and therefore, there must also be some Skolem constant b_1, s.t. $p_1(a,b_1)$ and $C_1(b_1)$ are 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, $\mathcal{I}' \models p_1(a,b_1) \wedge C_1(b_1)$. Similarly, we define $b_i^{\mathcal{I}'} = \delta_i$ for all $i \in [2,k]$, s.t. $\mathcal{I}' \models p_{i-1}(b_{i-1}, b_i) \wedge C_i(b_i)$, 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 the 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'. Our 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 $C \in \mathsf{N}^C_{\mathcal{K}}$, from which directly follows that for all $C(t)$ in $\mathcal{Q}$, $\sigma'(t) \in C^{\mathcal{I}}$. Similarly, $p^{\mathcal{I}'} = p^{\mathcal{I}}$ for all $p \in \mathsf{N}^P_{\mathcal{K}}$ and $\langle \sigma'(t_1), \sigma'(t_2) \rangle \in p^{\mathcal{I}}$ for all $p(t_1, t_2)$ 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. 3), which contradicts our hypothesis. □

Alg. 1 requires reasoning only to satisfy simple existential restrictions on individuals, that is, expressions of the form $\exists p.C(a)$ (see line 4). The latter is not a complex query and, in fact, it is not even a BCQ (Def. 3 excludes existential restrictions). Proving that $\mathcal{K} \models \exists p.C(a)$, 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, the "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 $(\mathsf{N}^I_{\mathcal{K}}.\mathsf{N}^R_{\mathcal{K}}.\mathsf{N}^C_{\mathcal{K}})^n$ 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 our experiments, rules in $\mathcal{K}$ have at most 5 variables. This allows us to retain tractability, as stated in the following theorem.

Theorem 2. *Let $\mathcal{K}$ be a knowledge base as per Def. 1, such that for all rule $B \rightarrow H$ in $\mathcal{K}$, B has at most n variables. Query answering for $\mathcal{K}$ 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. 1 is in the DL fragment of ELP, for which satisfiability and classification are polynomial [22]. 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 we have already mentioned, all the logics we are considering in this paper are standardized by OWL, the reference language for Web ontologies. There exist numerous commercial solutions for RDF storage that include standard OWL reasoning, among others 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 [42].

OWL reasoners are designed for expressive DL axioms, most of which being out of the scope of this paper 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 and load them into deductive database systems like RDFox [43] or XSB[6] [44]. In contrast to RDF store, however, most deductive database systems are prototypical or dedicated to research. In the following section, we discuss concrete implementation details and experiments.

4. Experiments

The semantic discovery framework we have defined in theoretical terms (Def. 7) relies on a discovery agent that first collects available TDs 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. We call this discovery agent a Thing Directory (TDir).

[4] http://graphdb.ontotext.com/
[5] https://www.stardog.com/
[6] if used in its Datalog flavor, with tabling.

Our TDir implementation offers a registration Web interface designed after the IETF Resource Directory specification[7] and backed by an RDF store (GraphDB). TDs 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[8]. The latter targets constrained environments by providing concise serializations. To a lesser extent, our TDir implementation is inspired from the early Hypercat specification, a Web resource catalogue with RDF-like annotations[9], with the difference that our implementation also provides a SPARQL interface with off-the-shelf OWL reasoning. The source code of our TDir can be found online[10].

We designed two experiments with TDir-based semantic discovery in different domains of application of WoT: BA and more precisely the Heating, Ventilation and Air Conditioning (HVAC) domain, where our use case is a sensor network setup provided by Intel Labs, and Industrial Control Systems (ICS) where our use case is centered on a water management plant model used in diverse scenarii at Siemens. We present both use cases next.

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[11]. It consists in 54 wireless devices called *motes* with as little as 8kB RAM, deployed homogeneously over an entire floor of the Intel Labs building (Fig. 3). All devices have identical capabilities. They can measure temperature, humidity and illuminance in various zones of the office: meeting rooms, closed offices, open space, entrance hall and kitchen.

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, we identified the following system 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 in HVAC (high localized gradient values).

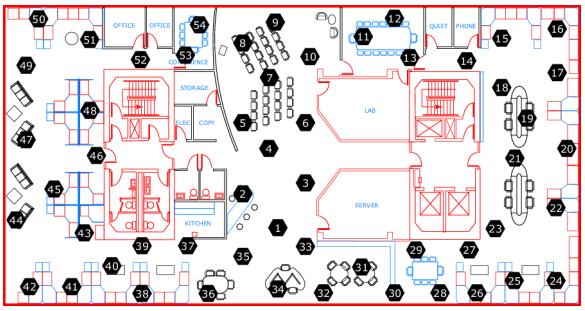


Fig. 3. Map of the Intel Labs sensor network (Source: Intel Labs)

If the system included actuators and sensors of different kinds, the range of possible interactions would be significantly wider. Yet, this particular use case offers us 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, we assume 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. For instance, the IFC information model, which is the basis for ifcOWL, is used by architects and civil engineers throughout the design of a building. It is part of what is commonly called a Building Information Model (BIM).

Example 13. *Below is an excerpt of the BIM we used for this experiment. We first defined a small class hierarchy for the Intel Labs office.*

$$\texttt{Wall}(x) \rightarrow \texttt{Solid}(x).$$

$$\texttt{Space}(x) \rightarrow \texttt{Air}(x).$$

$$\texttt{Space}(x) \rightarrow \texttt{Zone}(x).$$

$$\texttt{Hall}(x) \rightarrow \texttt{Space}(x).$$

$$\texttt{ConferenceRoom}(x) \rightarrow \texttt{Space}(x).$$

We then asserted the adjacency relations between spaces and separating walls (if any).

$$\texttt{Hall}(\texttt{hall}).$$

[7] https://datatracker.ietf.org/doc/draft-ietf-core-resource-directory/
[8] https://tools.ietf.org/html/rfc6690
[9] http://hypercat.io/
[10] https://github.com/thingweb/thingweb-directory/
[11] http://db.csail.mit.edu/labdata/labdata.html

$$\texttt{ConferenceRoom}(\texttt{room}).$$
$$\texttt{Wall}(\texttt{leftWall}).$$
$$\texttt{Wall}(\texttt{rightWall}).$$
$$\texttt{adjacentElement}(\texttt{hall}, \texttt{leftWall}).$$
$$\texttt{adjacentElement}(\texttt{room}, \texttt{leftWall}).$$
$$\texttt{adjacentElement}(\texttt{room}, \texttt{rightWall}).$$

The rest of our knowledge base for this use case is similar to what we gave in Ex. 11 (with instances of `TemperatureSensor` only). The complete RDF documents can be found online[12]. We ran semantic discovery for the CQ we provided earlier (after introducing the generic property `relatesToProperty`), for which we obtain the graph of interactions shown on Fig. 4. What is immediately identifiable on this figure are the three complete subgraphs that dominate the distribution of edges. Each represent devices observing 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 with any other device in that zone in order to detect anomalies on the measured temperature/humidity. Other subgraphs of size 1, 2 and 3 represent isolated devices located in meeting rooms or closed offices.

What this graph reveals is the high imbalance in terms of coverage of the physical space: if one of the isolated devices fails, an important part of the office is not 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. We plotted this data on Fig. 5. Over the course of the experiment, more and more devices go offline, most likely because of an empty battery, until the percentage of online devices reaches 0% on April 3rd (red area). As a comparison, we computed a graph of interactions at every point in time and computed the percentage of remaining connected subgraphs compared to the original graph. That is, we computed 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 were still present although 20% of devices went offline. Then, until April 1st, although only 10% of devices were still online, we still had 50% of physical coverage.

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

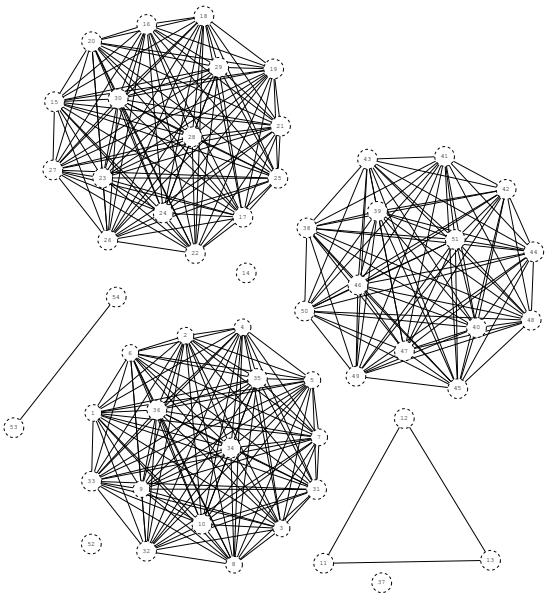


Fig. 4. Graph of interactions for the Intel Labs sensor network

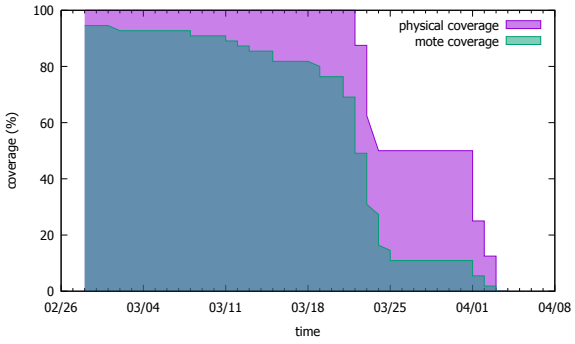


Fig. 5. Mote coverage vs. physical (system) coverage of the Intel Labs office

4.2. Use Case: Water Management Plant

The Intel Labs setup was highly homogeneous and exploratory insofar that we assumes future WoT systems will have similar features. In contrast, the other setup we have 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-controllers (MCUs) acting as WoT servients with IP connectivity. An overview of the workstation is provided on Fig. 6.

Fig. 6. Water management workstation with NodeMCU micro-controllers (Source: FESTO)

These micro-controllers are ESP8266 (64kB RAM, 80 MHz), they all embed a TD using the μRDF store, a lightweight RDF store designed for constrained devices [45]. We recently showed that a combination of a JSON-LD compacted representation of RDF triples and binary JSON encoding performs better than the state-of-the-art for small datasets like TDs. RDF triples are first processed using JSON-LD compaction with a context document designed for conciseness and then encoded in the EXI4JSON format, a format based on the Efficient XML Interchange (EXI) format [46]. These TDs are defined in terms of SOSA, SSN and eCl@ss. We slightly extended eCl@ssOWL for the needs of our experiment. Essentially, we provided an alignment with SSN and gave the physical quantities sensors and actuators relate to.

Example 14. *We distinguished between automation devices (sensors/actuators) and other kind of equipment. We define the former as* `ssn:Systems` *and the latter as* `sosa:Platforms`*, as follows:*

$$\texttt{WaterTank}(x) \rightarrow \texttt{Platform}(x).$$

$$\texttt{PlasticPipe}(x) \rightarrow \texttt{Platform}(x).$$

$$\texttt{Pump}(x) \rightarrow \texttt{Actuator}(x).$$

$$\texttt{Pump}(x) \rightarrow \exists \texttt{actsOnProperty}.\texttt{FlowRate}(x).$$

$$\texttt{PneumaticValve}(x) \rightarrow \texttt{Actuator}(x).$$

$$\texttt{PneumaticValve}(x) \rightarrow \exists \texttt{actsOnProperty}.\texttt{FlowRate}(x).$$

$$\texttt{FloatSwitch}(x) \rightarrow \texttt{Sensor}(x).$$

$$\texttt{FloatSwitch}(x) \rightarrow \exists \texttt{observes}.\texttt{Height}(x).$$

$$\texttt{UltrasonicSensor}(x) \rightarrow \texttt{Sensor}(x).$$

$$\texttt{UltrasonicSensor}(x) \rightarrow \exists \texttt{observes}.\texttt{Height}(x).$$

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

Given rules like Ex. 7 about the deformability of fluids, we 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, we assume that some factory plant or circuit description is available as DL assertions, after transformation from another machine-readable format. In total, we identified five systems that can be 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.

We ran semantic discovery for the same CQ in the Intel Labs setup (with `relatesToProperty`). We obtained 8 edges in the graph of interactions (computation time below 1s). All 5 compound systems can be instantiated and 'Water Circulation' has two solutions, while 'Valve Control' has three solutions. The whole graph of interaction is showed in Fig. 7. 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 we described in the previous section. Here, one of the nodes has a degree of 7, for a maximum degree of 16. If the servient is decommissioned and removed from the network, half of the discovered systems would stop functioning.

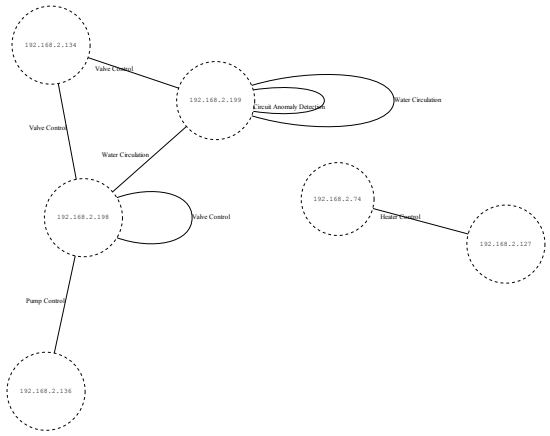


Fig. 7. Interaction graph resulting from semantic discovery on a water management workstation (servients are identified by their IP address)

In addition, we looked at the exchange that takes place between the TDir and servients. In a mediated scenario, the only exchange required for discovery is the registration of each TD on the directory. Registration can either happen by requests sent by individual servients to the TDir or by a single request broadcast by the TDir to all μRDF store instances on the network. In the former case, TDs are put in the request payload while in the latter case, they are in the response payload. In Fig. 8a, we show the size of the payload each servient sends.

In a peer-to-peer scenario, in addition to gathering servients' TDs, the TDir must update them, as per Def. 8. Updates are sent by the TDir to each μRDF store, with statements to add in the request payload. Payload sizes for updates are shown in Fig. 8b. One can note that in most cases, the update size is comparable to the size of the original TD. Both TDs and updates, except one, fit in a single CoAP block (of maximum size 1024 bytes), which represents no technical challenge for MCUs like the ESP8266.

All RDF files and results shown in this paper are available online[13].

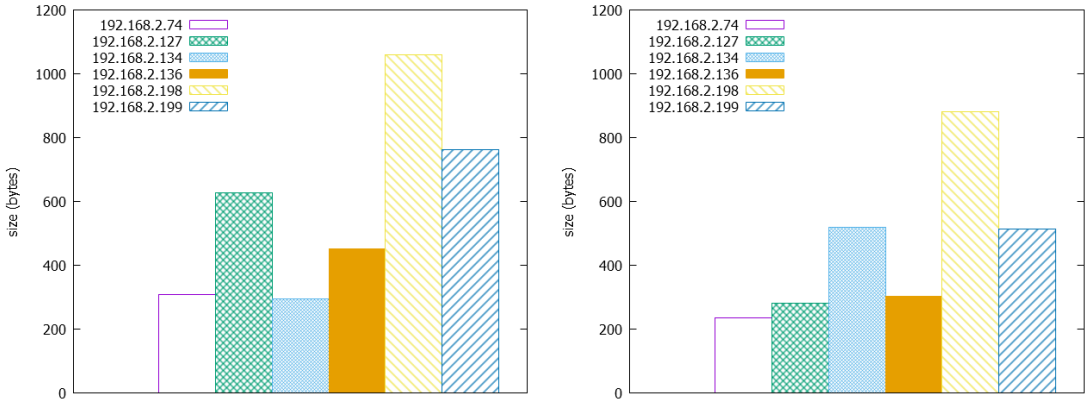
5. Conclusion

The last aspect of our experiment on the ICS setup shows that knowledge-base intelligent systems can be designed using WoT concepts and current technologies like CoAP and EXI4JSON. Moreover, both experiments in the HVAC and ICS domains show how WoT systems can be adaptable, an important characteristic of intelligence [47]: from the graphs of interaction we computed for both systems (Figs. 4 & 7), one can see that some tasks could be implemented in different ways by looking at the degree of vertices (servients).

Another kind of adaptibility, inherent to knowledge-based systems, is the ability of the system to change context by re-running semantic discovery with the same TDs but a different CBox. For instance, some conference rooms are reconfigurable to suit the needs of different kinds of events. In this context, simply changing the BIM of the building yields another graph of interactions, without having to reconfigure the servients. Intuitively, one can observe that the capabilities of a WoT system are directly related to its level of knowledge with respect to itself (via a collection of TDs) and its environment (via other ontological axioms).

Beyond semantic discovery, there are other reasoning tasks of interest given the abstraction for WoT interactions we introduce in this paper (that is, the characterization of interactions with CQs with existentially quantified variables). For instance, given a set of interactions taking place in a running system, one can compute the associated knowledge base explaining these interactions to e.g. detect incomplete axioms or detect inconsistencies in the system. These tasks are a potential line of research for future work.

[13] https://github.com/vcharpenay/urdf-store-exp



(a) Thing Description (b) Update ABox

Fig. 8. Size of knowledge bases exchanged between servients and Thing Directory, encoded in EXI4JSON (servients are identified by their IP address)

On another level, the feasability of our approach depends on whether the kind of axioms we have introduced in Sec. 3.1 can be defined at a large scale. Our observation that existential restrictions and equality relations between individuals were under-axiomatized can be generalized to most Web ontologies. As a consequence, future work also includes the engineering of a dedicated ontology for physical bodies and their properties, to fill the gap between existing models, like OM and BOT, and the level of knowledge required in WoT systems. We believe that most thermodynamic, optical, mechanical and geometrical properties of physical bodies can be axiomatized in terms of rules with a reasonable effort. What remains open, however, is how such an axiomatization can be combined with numeric models (like thermic simulation from a BIM).

References

[1] D. Guinard and V.M. Trifa, Towards the Web of Things: Web Mashups for Embedded Devices, in: *Proceedings of WWW (International World Wide Web Conferences)*.

[2] S. Poslad, *Ubiquitous Computing: Smart Devices, Environments and Interactions*, John Wiley & Sons, Inc. ISBN 978-0-470-77944-6.

[3] K. Kazuo, M. Kovatsch and U. Davuluru, Web of Things (WoT) Architecture. https://www.w3.org/TR/wot-architecture/.

[4] I. Toma, E. Simperl and G. Hench, A joint roadmap for Semantic technologies and the Internet of Things.

[5] S. Kaebisch and T. Kamiya, Web of Things (WoT) Thing Description. https://www.w3.org/TR/wot-thing-description/.

[6] V. Charpenay, S. Käbisch and H. 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*, CEUR-WS.org.

[7] T. Kindberg and S. Hawke, The 'tag' URI Scheme. https://tools.ietf.org/html/rfc4151.

[8] P. Saint-Andre and J. Klensin, Uniform Resource Names (URNs). https://tools.ietf.org/html/rfc8141.

[9] S. Farrell, D. Kutscher, C. Dannewitz, B. Ohlman, A. Keranen and P. Hallam-Backer, Naming Things with Hashes. https://tools.ietf.org/html/rfc6920.

[10] D. Guinard, Upgrading the Barcode to the Web: GS1 Digital Link. https://evrythng.com/upgrading-the-barcode-to-the-web-gs1-digital-link/.

[11] P. Pauwels and W. Terkaj, EXPRESS to OWL for construction industry: Towards a recommendable and usable ifcOWL ontology **63**, 100–133, ISSN 09265805. doi:10.1016/j.autcon.2015.12.003.

[12] B. Bharathan, A.A. Bhattacharya, G. Fierro, J. Gao, J. Gluck, D. Hong, A. Johansen, J. Koh, Y. Agarwal, M. Berges, D. Culler, R. Gupta, M. Baun Kjærgaard, J. Ploennigs and K. Whitehouse, Brick v1.0 - Towards a Unified Metadata Schema For Buildings.

[13] M.H. Rasmussen, P. Pauwels, M. Lefrançois, G.F. Schneider, C.A. Hviid and J. Karlsh, Recent changes in the Building Topology Ontology, in: *LDAC2017 - 5th Linked Data in Architecture and Construction Workshop*.

[14] A. Haller, K. Janowicz, S. Cox, D. Le Phuoc, K. Taylor and M. Lefrançois, Semantic Sensor Network Ontology. https://www.w3.org/TR/vocab-ssn/.

[15] R. Hajo, v.A. Mark and T. Jan, Ontology of units of measure and related concepts, 3–13, ISSN 1570-0844. doi:10.3233/SW-2012-0069.

[16] M. Hepp and A. Radinger, eClassOWL - The Web Ontology for Products and Services. http://www.heppnetz.de/projects/eclassowl/.

[17] R.V. Guha, D. Brickley and S. Macbeth, Schema.org: evolution of structured data on the web **59**(2), 44–51, ISSN 00010782. doi:10.1145/2844544.

[18] J. Leighton and R. Sternberg, *The nature of reasoning*, Cambridge University Press. ISBN 978-0-521-00928-7.

[19] P.J. Hayes and P. Patel-Schneider, RDF 1.1 Semantics, W3C. http://www.w3.org/TR/rdf11-mt/.

[20] A. Hogan, Skolemising Blank Nodes while Preserving Isomorphism, ACM Press, pp. 430–440. ISBN 978-1-4503-3469-3. doi:10.1145/2736277.2741653. http://dl.acm.org/citation.cfm?doid=2736277.2741653.

[21] A. Hogan, Canonical Forms for Isomorphic and Equivalent RDF Graphs: Algorithms for Leaning and Labelling Blank Nodes **11**.

[22] M. Krötzsch, S. Rudolph and P. Hitzler, ELP: Tractable Rules for OWL 2, in: *The Semantic Web - ISWC 2008*, A. Sheth, S. Staab, M. Dean, M. Paolucci, D. Maynard, T. Finin and K. Thirunarayan, eds, Springer Berlin Heidelberg, pp. 649–664. ISBN 978-3-540-88564-1.

[23] F. Baader, S. Brandt and C. Lutz, Pushing the EL envelope, in: *IJCAI*, Vol. 5, pp. 364–369.

[24] F. Baader, S. Brandt and C. Lutz, *Pushing the EL envelope further*.

[25] S. Rudolph, Foundations of Description Logics, in: *Reasoning Web. Semantic Technologies for the Web of Data*, A. Polleres, C. d'Amato, M. Arenas, S. Handschuh, P. Kroner, S. Ossowski and P. Patel-Schneider, eds, Springer Berlin Heidelberg, pp. 76–136. ISBN 978-3-642-23031-8 978-3-642-23032-5.

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

[27] F. Gasse, U. Sattler and V. Haarslev, Rewriting Rules into SROIQ Axioms, in: *Proceedings of the DL 21st International Workshop on Description Logics (DL2008)*, Vol. 353, CEUR-WS.org.

[28] S. Ceri, G. Gottlob and L. Tanca, What you always wanted to know about Datalog (and never dared to ask) **1**(1), 146–166, ISSN 10414347. doi:10.1109/69.43410.

[29] B.N. Grosof, I. Horrocks, R. Volz and S. Decker, Description logic programs, 48. doi:10.1145/775152.775160.

[30] A. Calì, G. Gottlob and T. Lukasiewicz, A general Datalog-based framework for tractable query answering over ontologies **14**, 57–83, ISSN 15708268. doi:10.1016/j.websem.2012.03.001.

[31] M. Krötzsch, S. Rudolph and P. Hitzler, Conjunctive Queries for a Tractable Fragment of OWL 1.1, in: *The Semantic Web*, Vol. 4825, K. Aberer, K.-S. Choi, N. Noy, D. Allemang, K.-I. Lee, L. Nixon, J. Golbeck, P. Mika, D. Maynard, R. Mizoguchi, G. Schreiber and P. Cudré-Mauroux, eds, Springer Berlin Heidelberg, pp. 310–323. ISBN 978-3-540-76297-3 978-3-540-76298-0. doi:10.1007/978-3-540-76298-$0_2$3.

[32] D. Calvanese, G.D. Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Data complexity of query answering in description logics **195**, 335–360, ISSN 0004-3702. doi:https://doi.org/10.1016/j.artint.2012.10.003.

[33] D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini and R. Rosati, Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family **39**(3), 385–429, ISSN 0168-7433, 1573-0670. doi:10.1007/s10817-007-9078-x.

[34] C. Gutierrez, C. Hurtado and A.O. Mendelzon, Foundations of semantic web databases, ACM Press, p. 95. ISBN 978-1-58113-858-0. doi:10.1145/1055558.1055573.

[35] A.C. KAKAS, R.A. KOWALSKI and F. TONI, Abductive Logic Programming **2**, 719–770. doi:10.1093/logcom/2.6.719.

[36] Z. Wang, M. Chitsaz, K. Wang and J. Du, Towards Scalable and Complete Query Explanation with OWL 2 EL Ontologies, 743–752. doi:10.1145/2806416.2806547.

[37] S. Klarman, U. Endriss and S. Schlobach, ABox Abduction in the Description Logic $\mathcal{ALC}$ **46**, 43–80. doi:10.1007/s10817-010-9168-z.

[38] J. Du, K. Wang and Y.-D. Shen, A Tractable Approach to ABox Abduction over Description Logic Ontologies., in: *AAAI*, pp. 1034–1040.

[39] M. Gavanelli, E. Lamma, F. Riguzzi, E. Bellodi, R. Zese and G. Cota, Reasoning on Datalog$^{\pm}$ Ontologies with Abductive Logic Programming, 65–93, ISSN 18758681. doi:10.3233/FI-2018-1658.

[40] V. Charpenay, S. Käbisch and H. Kosch, A Framework for Semantic Discovery on the Web of Things, 147–162, ISSN 1868-1158. doi:10.3233/978-1-61499-894-5-147.

[41] S. Lohmann, S. Negru, F. Haag and T. Ertl, Visualizing Ontologies with VOWL **7**(4), 399–419. doi:10.3233/SW-150200.

[42] T.W.S.W. Group, SPARQL 1.1 Overview, W3C. https://www.w3.org/TR/2013/REC-sparql11-overview-20130321/.

[43] Y. Nenov, R. Piro, B. Motik, I. Horrocks, Z. Wu and J. Banerjee, RDFox: A Highly-Scalable RDF Store, in: *The Semantic Web - ISWC 2015*, Vol. 9367, M. Arenas, O. Corcho, E. Simperl, M. Strohmaier, M. d'Aquin, K. Srinivas, P. Groth, M. Dumontier, J. Heflin, K. Thirunarayan and S. Staab, eds, Springer International Publishing, pp. 3–20. ISBN 978-3-319-25009-0 978-3-319-25010-6.

[44] K. Sagonas, T. Swift and D.S. Warren, XSB as an efficient deductive database engine **23**(2), 442–453, ISSN 01635808. doi:10.1145/191843.191927.

[45] V. Charpenay, S. Käbisch and H. Kosch, μRDF Store: Towards Extending the Semantic Web to Embedded Devices, in: *The Semantic Web: ESWC 2017 Satellite Events*, Vol. 10577, E. Blomqvist, K. Hose, H. Paulheim, A. ŁLawrynowicz, F. Ciravegna and O. Hartig, eds, Springer International Publishing, pp. 76–80. ISBN 978-3-319-70406-7 978-3-319-70407-4. doi:10.1007/978-3-319-70407-$4_1$5.

[46] V. Charpenay, S. Käbisch and H. Kosch, Towards a Binary Object Notation for RDF, in: *Proceedings of the 15th Extended Semantic Web Conference (ESWC)*, Springer.

[47] S.J. Russell and P. Norvig, *Artificial intelligence: a modern approach*, 3rd ed edn, *Prentice Hall series in artificial intelligence*, Prentice Hall. ISBN 978-0-13-604259-4.

[48] P. Leach, A Universally Unique IDentifier (UUID) URN Namespace, IETF. https://tools.ietf.org/html/rfc4122.

[49] M. Rodríguez-Muro and D. Calvanese, High performance query answering over DL-Lite ontologies, in: *Proceedings of the 13th International Conference on Principles of Knowledge Representation and Reasoning (KR 2012)*.