A Framework for Semantic Discovery on the Web of Things*

Victor Charpenay[1,2], Sebastian Käbisch[1], and Harald Kosch[2]

[1] Siemens AG — Corporate Technology
victor.charpenay@siemens.com, sebastian.kaebisch@siemens.com
[2] Universität Passau — Fakultät für Informatik und Mathematik
harald.kosch@uni-passau.de

Abstract. This paper addresses the problem of discovering Web of Things (WoT) agents, also known as servients, that can interact in a mediated or peer-to-peer fashion to form compound systems. We develop a framework that relies on the W3C Thing Description (TD) and Semantic Sensor Network (SSN) ontologies, which provide semantics for the physical world entities WoT servients are associated with.
We formulate the problem of WoT discovery as an abductive reasoning problem over knowledge bases expressed in terms of TD and SSN concepts, where new semantic relationships between existing systems lead to the creation of other, larger systems. We then address the specific case of $\mathcal{EL}^{++}$ knowledge bases, a fragment of Description Logic, for which we leverage the mathematical framework of Abductive Logic Programming to provide a concrete algorithm for abduction that terminates in polynomial time.
We illustrate the feasability of our approach on an experimental industrial workstation, equipped with micro-controllers capable of storing and exchanging RDF data in binary format (μRDF store with EXI4JSON serialization).

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 [10]. 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. In this context, Semantic Web technologies should be used to represent physical world entities and provide a consistent view on cyber-physical systems [24].

These semantically described WoT systems may be seen as intelligent systems, as autonomy is regarded as an important characteristic of intelligence [23]. But more importantly, WoT systems are multi-agent systems (MAS), as are other ubiquitous and pervasive systems [21]. A WoT system is first described by the

* This work will be published as part of the book "Emerging Topics in Semantic Technologies. ISWC 2018 Satellite Events. E. Demidova, A.J. Zaveri, E. Simperl (Eds.), ISBN: 978-3-89838-736-1, 2018, AKA Verlag Berlin".

interactions that take place between WoT agents, acting either as clients or as servers. In most cases, interactions take place either between one client and several servers, or between devices that alternatively act as client and server [16]. 'Servient' is a generic term that designates either clients, servers, or devices that can be both. In the MAS terminology, one speaks of *mediated* interaction (the client playing the role of a mediator) and *peer-to-peer* interaction.

The problem we address in this paper is that of automatically discovering WoT agents (servients) for which a potential interaction in a MAS exists. That is, we aim at building a 'graph of interactions' between servients, potentially at Web scale. For instance, in a mediated scenario, a WoT client could execute the following procedure to empty a water tank by opening a valve when the tank is full:

1. send `GET coap://{IP1}/watertank/overflow`
2. if returned value is `true`,
   (a) then send `PUT coap://{IP2}/valve/status 'true'`
3. sleep for 1s and go to step 1.

Here, semantic discovery consists in finding the pair of servients (`IP1`, `IP2`)—an edge in the graph of interactions—, such that one exposes a boolean value representing the detection of overflow in a water tank and the other exposes a writable boolean value to control a valve in a water circulation circuit.

In this paper, we address the problem of WoT discovery by relying on Semantic Web technologies. More precisely, we express possible servient interactions in terms of the relations between the physical world entities they are associated with. In the remainder of the paper, we speak of 'semantic discovery'. The paper is organized as follows: we first introduce the semantic models one can use to associate servients with physical world entities, most importantly the Semantic Sensor Network (SSN) ontology; then, we formulate the problem of semantic discovery as an abductive reasoning task on the SSN class of `System`; we then present an algorithm for $\mathcal{EL}^{++}$ ontologies by leveraging the mathematical framework of Abductive Logic Programming; we finally present experimentations on an prototype workstation equipped with micro-controllers, and conclude on the provided results.

2 Related Work

Several discovery mechanisms can coexist in the Web of Things [16]. However, the most common one so far has been directory-based discovery. That is, WoT servients register themselves on a central directory by providing meta-data about their capabilities and the Web resources they expose. Different proposals have been made for a standardized interface to such a directory. Examples include Hypercat[3] or the IETF Resource Directory specification[4]. Hypercat specifies a

[3] http://hypercat.io/
[4] https://datatracker.ietf.org/doc/draft-ietf-core-resource-directory/

hypermedia catalogue, on which plain Web resources can be registered and annotated with RDF-like statements. Annotations can either be used to connect resources with each others or to connect them with semantic resources from an external knwoledge base, e.g. an RDF store. The IETF Resource Directory provides similar functionalities but it relies on the IETF CoRE Link format for resource annotations, which targets constrained environments by providing concise serializations[5]. Both proposals provide simple query functionalities (key/value filtering).

The Eclipse Thingweb project, that implements W3C standards for the Web of Things, includes an implementation of a 'Thing Directory'[6], inspired by the IETF specification but allowing arbitrary RDF annotations, like Hypercat. WoT servients can register so-called 'Thing Descriptions' (TDs) on a Thing Directory. A TD is an RDF document that links physical world entities (instances of the concept of `Thing`) to Web resources [14]. In the TD model, a distinction is made between 'semantic resources' and other Web resources [3]. Semantic resources, in that context, are resources that act as proxies for physical world entities: they provide a representation of these entities in a formal language. This distinction helps identifying independently a thing itself and the Web resource that provides data about this thing. For example, if `coap://example.org/valve` provides JSON data about the valve, the valve itself, as a physical asset, is proxified by another (semantic) resource, identified e.g. by a blank node or the arbitrary UUID `urn:uuid:17b4b33d-91a9-362a-b3f6-087d57fcdc47`.

This problem of identity of physical world entities is well-known in the Semantic Web community and best practices have emerged to distinguish semantic resources from other Web resources: one should exclusively use hash URIs or HTTP 303 redirection to identify them[7]. However, in the context of WoT, uniquely identifying all physical world entities involved in a system is a challenge. Even in very simple cases, there is no obvious way to identify inanimate objects (like the water tank in our introductory example). A convention for manufactured products could be to use their serial number. Another convention could be to use arbitrary identifiers encoded in RFID tags. However, even if one assumes there exists a widely agreed upon addressing mechanism to unambiguously give an IRI to any physical world entity, this kind of approach requires the deployment of a vast and heterogeneous network to combine near-field communication (to expose a product serial number) with wired local connectivity and other wireless technologies. This high human integration effort hinders, to some extent, the vision of WoT as a network of autonomous systems.

An alternative to this approach is to leverage the model theoretical semantics of the TD model and Semantic Web technologies. By means of reasoning, the existence of certain physical world entities can be inferred from statements contained in TD documents. Indeed, TDs can contain arbitrary statements about `Thing`s, along with the links between these `Thing`s and Web resources, by using

[5] https://tools.ietf.org/html/rfc6690
[6] https://projects.eclipse.org/projects/iot.thingweb
[7] https://www.w3.org/TR/cooluris/#semweb

any RDF vocabulary. Numerous already existent vocabularies can be used, all of them being grounded in Descirfpation Logic (DL). The most important one is the SSN ontology [11,13,12]. SSN extends the lightweight Sensor, Observation, Sample and Actuator (SOSA) ontology, defined in the same W3C standard; it is itself extended by the Smart Energy Aware SystemsLaisse (SEAS) set of ontologies[8] [19]. Another important vocabulary for WoT is the Smart Appliance Reference (SAREF) ontology, standardized by ETSI [1]. Formal alignments between SSN, SEAS and SAREF have been recently proposed by the Semantic Web community [19,20].

WoT servients exposing TDs annotated with SSN, SEAS and SAREF form a promising basis for semantic discovery. A typical TD would expose a `Thing` as an SSN `System` or a SOSA `FeatureOfInterest`, whose `Properties` are exposed via Web resources. Interacting with the servient corresponds to either a SOSA `Observation` or a SOSA `Actuation`. SAREF defines more specific concepts like `TemperatureSensor` and `Temperature` (specializations of `System` and `Property`, respectively). Semantic discovery would then consist in performing automated reasoning on the statements from various TDs registered on a Thing Directory.

DL concepts for industrial equipment would be required for fine-grain discovery in cyber-physical systems. In our introductory example, concepts for water tanks and valves can be found in the industrial standard eCl@ss[9]: `27309210` (reservoir/tank for hydraulics), `27292200` (pneumatic valve), `27292300` (proportional valve). It is also possible to combine SSN with ontologies for physical quantities and units like QUDT[10] or OM[11].

In the following, we formalize the problem of semantic discovery in the Web of Things in terms of DL reasoning.

3 Problem Statement

Every WoT servient i exposes in the form of a TD a set of logical assertions $\mathcal{A}_i$ (called an ABox) with respect to a shared set of axioms $\mathcal{C}$ (called a CBox). We denote $\mathcal{A}$ the ABox $\bigcup_i \mathcal{A}_i$ and $\mathcal{KB}$ the knowledge base defined as $\mathcal{A} \cup \mathcal{C}$.

We denote $\mathsf{N}^I_{\mathcal{A}}$ the set of individuals (i.e. Web resources) in $\mathcal{A}$. As argued in Sec. 2, we do not assume that there exists a global adressing mechanism to give an IRI to any physical world entity. As a consequence, $a \in \mathsf{N}^I_{\mathcal{A}_i}$ and $b \in \mathsf{N}^I_{\mathcal{A}_j}$ $(i \neq j)$ may designate the same entity, i.e. the *unique name assumption* does not generally apply to WoT knowledge bases. But it also means that entities potentially useful for discovery may not be represented in the knowledge base at all. In this context, deductive reasoning would not be sound if potential servient interactions involve entities whose existence can neither be asserted nor inferred. However, in many cases, their existence can reasonably be simply assumed, so that deduction leads to the expected conclusions. We formalize the problem of

[8] https://w3id.org/seas/
[9] http://www.heppnetz.de/projects/eclassowl/
[10] http://qudt.org/
[11] http://www.wurvoc.org/vocabularies/om-1.8/

semantic discovery in the Web of Things as an *abduction* problem, to compute such assumptions.

Definition 1. *[22] Abduction consists in finding a knowledge base $\mathcal{KB}'$, such that, for a knowledge base $\mathcal{KB}$ and an axiom α, it holds that $\mathcal{KB} \cup \mathcal{KB}' \models \alpha$ and $\mathcal{KB} \not\models \alpha$.*

In our case, we aim at discovering new possible systems by making various WoT servients interact. Semantic discovery can therefore be formulated as the abduction of role assertions between individuals of two or more distinct ABoxes and possibly fresh individuals, such that new instances of `System` are created in the overall knowledge base.

Definition 2. *WoT semantic discovery on a knowledge base $\mathcal{KB} = \bigcup_i \mathcal{A}_i \cup \mathcal{C}$ is the abductive reasoning task where $\alpha = \texttt{System}(a)$ and $\mathcal{KB}'$ is a set of role assertions for which there exists i, j, such that $\mathsf{N}^I_{\mathcal{KB}'} \cap \mathsf{N}^I_{\mathcal{A}_i} \neq \emptyset$ and $\mathsf{N}^I_{\mathcal{KB}'} \cap \mathsf{N}^I_{\mathcal{A}_j} \neq \emptyset$.*

We do not consider class assertions in $\mathcal{KB}'$ to avoid trivial solutions like $\mathcal{KB}' = \{\texttt{System}(a)\}$, where a is a fresh individual. In practice, one can exploit the mereologic relation `hasSubSystem` to create a new system as the combination of two or more sub-systems.

Example 1. Let us define a knowledge base for our introductory example, as would be available on a Thing Directory:

$$
\begin{aligned}
\mathcal{A}_1 = \{&\texttt{27292200}(\texttt{valve}),\\
&\texttt{href}(\texttt{valve}, \texttt{coap://192.168.0.1/valve/status})\}\\
\mathcal{A}_2 = \{&\texttt{27273201}(\texttt{sensor}),\\
&\texttt{href}(\texttt{sensor}, \texttt{coap://192.168.0.1/tank/overflow}),\\
&\texttt{27273201}(\texttt{tank}),\\
&\texttt{isHostedBy}(\texttt{sensor}, \texttt{tank})\}\\
\mathcal{C} = \{&\texttt{ValveControlSystem} \sqsubseteq \texttt{System},\\
&\exists \texttt{hasSubSystem}.\texttt{27292200} \sqcap\\
&\exists \texttt{hasSubSystem}.(\texttt{27273201} \sqcap \exists \texttt{isHostedBy}.\texttt{27273201})\\
&\sqsubseteq \texttt{ValveControlSystem}\}
\end{aligned}
$$

Here, `27292200`, `27273201` and `27273201` are the eCl@ss concepts for pneumatic valve, level sensor, and water tank (respectively). We define here the role `href` as a direct link from a `Thing` instance to a Web resource, which is a simplification of the original TD model.

The following axioms are a solution to semantic discovery on the above knowledge base:

$$\mathcal{KB}' = \{\texttt{hasSubSystem}(a, \texttt{valve}), \texttt{hasSubSystem}(a, \texttt{sensor})\}$$

In the case where interaction between servients is mediated, the mediator can simply access the Thing Directory's knowledge base after it is added $\mathcal{KB}'$. However, in a peer-to-peer setup, local ABoxes (i.e. TDs stored on the servient themselves) 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. This relates to the notion of *explanation*.

Definition 3. *[22] An explanation for an axiom α is a knowledge base $\mathcal{E} \subseteq \mathcal{KB}$ such that $\mathcal{E} \models \alpha$ but for every subset $\mathcal{E}' \subset \mathcal{E}$, $\mathcal{E}' \not\models \alpha$.*

Any servient involved in a newly discovered system should be updated with a set of statements that would allow it to infer from its ABox alone the class assertion axiom used for discovery.

Definition 4. *Let $\mathcal{E}$ be an explanation over $\mathcal{KB} \cup \mathcal{KB}'$ for an axiom of the form* $\texttt{System}(a)$*. The update ABox of a WoT semantic discovery in a peer-to-peer system is the ABox $\mathcal{A}' = \bigcup_i \mathcal{A}'_i$, such that for all $\mathcal{A}_i$ where $\mathsf{N}^I_{\mathcal{A}_i} \cap \mathsf{N}^I_{\mathcal{E}} \neq \emptyset$, $\mathcal{A}'_i = \mathcal{E} \setminus \mathcal{A}_i$.*

Given Definition 4, it then holds that $(\mathcal{A}_i \cup \mathcal{A}'_i) \cup \mathcal{C} \models \texttt{System}(a)$. 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 $\mathcal{A}'_i \cap \mathcal{A}''_i = \emptyset$. It is indeed likely that servients enter the network at different times over the lifecycle of an industrial system.

Example 2. The update ABox for Example 1 is the following:

$$\mathcal{A}'_1 = \{\texttt{hasSubSystem}(a, \texttt{sensor}\}$$
$$\mathcal{A}'_2 = \{\texttt{hasSubSystem}(a, \texttt{valve}\}$$

One can note in Example 2 that the performed update is of little value as long as the valve has no knowledge of the Web resource to access the level sensor value (or as long as the sensor has no knowledge of the valve status update Web resource). To include them in their respective update ABox, one can modify $\mathcal{C}$ by adding restrictions on `href` to sub-concepts of `System`. In general, the value of semantic discovery depends on the axioms present in $\mathcal{C}$. We give more examples of axioms used for discovery in Sec. 5 (Example 3).

4 A Logic Programming Approach

In this paper, we focus on semantic discovery with $\mathcal{EL}^{++}$ knowledge bases, the only tractable fragment of DL. There exists other works addressing ABox abduction on other DL fragments [17,7] but we limited ourselves to $\mathcal{EL}^{++}$ for several reasons. First, tractability, and therefore predictability, is often a strict

requirement in the industry domain. Moreover, most axioms that are relevant in WoT knowledge bases are existential restrictions (the ones covered by $\mathcal{EL}^{++}$), as opposed to universal restrictions and cardinality constraints, which are often redundant with other forms of data validation like data schemas. Although the state-of-the-art includes a tractable algorithm for so-called *first-order rewritable* knowledge bases, some $\mathcal{EL}^{++}$ do not fulfil this requirement. In addition, $\mathcal{EL}^{++}$ reasoning remains tractable even in the case servients provide different CBoxes (which can happen in case e.g. of firmware update or device recommissioning).

Classification of $\mathcal{EL}^{++}$ knowledge bases can be computed in polynomial time, as first introduced by Baader et al [2]. In the following, we formalize Baader et al.'s algorithm in terms of logic programming, in order to formalize WoT semantic discovery (Definition 2) as an abductive logic programming (ALC) problem.

4.1 $\mathcal{EL}^{++}$ Classification

For an extensive documentation of the original classification algorithm and the semantics of $\mathcal{EL}^{++}$, we refer to Baader et al.'s technical report.

Definition 5. *[2] Let $\mathsf{BC}_\mathcal{C}$ be the set of concept names and nominal concepts of the form $\{a\}$ ($a \in \mathsf{N}_\mathcal{C}^I$) and $\mathsf{R}_\mathcal{C}$ be the set of role names in $\mathcal{C}$. Let $C_1, C_2 \in \mathsf{BC}_\mathcal{C} \cup \{\top\}$, $D \in \mathsf{BC}_\mathcal{C} \cup \{\top, \bot\}$ and $r, s, r_1, r_2 \in \mathsf{R}_\mathcal{C}$.*

$\mathcal{C}$ is an $\mathcal{EL}^{++}$ normal form if any axiom in $\mathcal{C}$ is one of:

$$C_1 \sqsubseteq D \qquad r \sqsubseteq s$$
$$C_1 \sqcap C_2 \sqsubseteq D \qquad r_1 \circ r_2 \sqsubseteq s$$
$$C_1 \sqsubseteq \exists r.C_2$$
$$\exists r.C_1 \sqsubseteq C_2$$

Any $\mathcal{EL}^{++}$ knowledge base can be emulated by a CBox $\mathcal{C}$ in normal form. Class assertions $C(a)$ are rewritten $\{a\} \sqsubseteq C$ and role assertions $r(a, b)$ are rewritten $\{a\} \sqsubseteq \exists r.\{b\}$. In the following, we assume all knowledge bases are in normal form.

Baader et al.'s algorithm consists in applying eleven completion rules to a normalized knowledge base $\mathcal{C}$, to construct a mapping S, such that for $C \in \mathsf{BC}_\mathcal{C}$, $S(C) \subseteq \mathsf{BC}_\mathcal{C} \cup \{\top, \bot\}$ and a mapping R, such that for $r \in \mathsf{R}_\mathcal{C}$, $R(r) \subseteq \mathsf{BC}_\mathcal{C} \times \mathsf{BC}_\mathcal{C}$. Deciding whether $\mathcal{C}$ entails a given class inclusion can then be decided by a single look-up of S or R. Briefly:

- if $D \in S(C)$, it implies that $\mathcal{C} \models C \sqsubseteq D$
- if $(C, D) \in R(r)$, it implies that $\mathcal{C} \models C \sqsubseteq \exists r.D$

In the present paper, we reformulate Baader et al.'s algorithm in terms of logic programming [8]. A logic program is a set of clauses of the form *head* $\leftarrow$ *body* where *head*, *body* are first-order logic (FOL) formulas with variables. We first define a mapping from $\mathcal{EL}^{++}$ constructs to FOL formulas.

Definition 6. *The mapping τ from DL constructs to FOL formulas is defined recursively, as follows:*

$$
\begin{aligned}
\tau(C), \tau(r), \tau(a) &\text{ map to variables} \\
\tau(D \in S(C)) &= \mathtt{s}(\tau(C), \tau(D)) \\
\tau((C, D) \in R(r)) &= \mathtt{r}(\tau(r), \tau(C), \tau(D)) \\
\tau(C \sqsubseteq D \in \mathcal{C}) &= \mathtt{subClassOf}(\tau(C), \tau(D))) \\
\tau(\{a\}) &= \mathtt{objectOneOf}(\tau(a)) \\
\tau(C \sqcap D) &= \mathtt{objectIntersectionOf}(\tau(C), \tau(D)) \\
\tau(\exists r.D) &= \mathtt{objectSomeValuesFrom}(\tau(r), \tau(D)) \\
\tau(r \sqsubseteq s \in \mathcal{C}) &= \mathtt{subObjectPropertyOf}(\tau(r), \tau(s)) \\
\tau(r_1 \circ r_2) &= \mathtt{objectPropertyChain}(\tau(r_1), \tau(r_2))
\end{aligned}
$$

One may note that our definition of τ associates DL axioms to plain formulas. This differs from classical logic programming embeddings of DL, which turn axioms into rules (e.g. with Datalog [9]). Given Definition 6, we can now define the original completion rules[12] as a logic program.

Definition 7. *The logic program P for $\mathcal{EL}^{++}$ classification is defined as follows:*

$$
\begin{aligned}
\mathit{true} &\leftarrow \tau(C \in S(C)) && \\
\tau(D \in S(C)) &\leftarrow \tau(C' \in S(C)) \wedge && \text{(CR1)} \\
&\quad\ \tau(C' \sqsubseteq D \in \mathcal{C}) && \\
\tau(D \in S(C)) &\leftarrow \tau(C_1 \in S(C)) \wedge && \text{(CR2)} \\
&\quad\ \tau(C_2 \in S(C)) \wedge && \\
&\quad\ \tau(C_1 \sqcap C_2 \sqsubseteq D \in \mathcal{C}) && \\
\tau((C, D) \in R(r)) &\leftarrow \tau(C' \in S(C)) \wedge && \text{(CR3)} \\
&\quad\ \tau(C' \sqsubseteq \exists r.D \in \mathcal{C}) && \\
\tau(E \in S(C)) &\leftarrow \tau((C, D) \in R(r)) \wedge && \text{(CR4)} \\
&\quad\ \tau(D' \in S(D)) \wedge && \\
&\quad\ \tau(\exists r.D' \sqsubseteq E \in \mathcal{C}) && \\
\tau(\bot \in S(C)) &\leftarrow \tau((C, D) \in R(r)) \wedge && \text{(CR5)} \\
&\quad\ \tau(\bot \in S(D)) && \\
\tau(S(D) \subseteq S(C)) &\leftarrow \tau(\{a\} \in S(C) \cap S(D)) \wedge && \text{(CR6)} \\
&\quad\ \tau(C \rightsquigarrow D) && \\
\tau((C, D) \in R(s)) &\leftarrow \tau((C, D) \in R(r)) \wedge && \text{(CR10)} \\
&\quad\ \tau(r \sqsubseteq s \in \mathcal{C}) &&
\end{aligned}
$$

[12] In the present work, we discard rules 7-9, as they relate to the out-of-scope notion of concrete domains. Moreover, rule 6 includes the $\rightsquigarrow$ notation which, although it is also embeddable in FOL, we do not develop here.

$$\tau((C,E) \in R(r_3)) \leftarrow \tau((C,D) \in R(r_1)) \wedge \tau((D,E) \in R(r_2)) \wedge \tau(r_1 \circ r_2 \sqsubseteq r_3 \in \mathcal{C}) \quad \text{(CR11)}$$

Deciding whether $D \in S(C)$ can then be reduced to derivation on the predicate $\mathtt{s}$. That is, given a set of clauses P' stating class and role inclusions, it is equivalent to proving $P \cup P' \vdash \mathtt{s}(\tau(C), \tau(D))$, for which standard resolution can be applied (forward or backward chaining).

4.2 $\mathcal{EL}^{++}$ Abduction

On this basis, it is possible to define abduction on $\mathcal{EL}^{++}$ knowledge bases as an Abductive Logic Programming (ALP) problem. ALP extends standard logic program resolution with hypothesis generation from abducible predicates and integrity constraint checking [15]. ALP can be used to provide explanations to deductive reasoning tasks or for knowledge assimilation. Here, we use ALP to compute an explanation $\mathcal{E}$ for WoT discovery (Definition 4) with the introduction of fresh individuals under certain constraints. The set of generated axioms $\mathcal{KB}'$ (Definition 2) then corresponds to the set of axioms in $\mathcal{E}$ with fresh individuals.

Proposition 1. *A set of axioms $\mathcal{E}$ is an explanation to the problem of WoT semantic discovery on an $\mathcal{EL}^{++}$ knowledge base $\mathcal{C}$ if and only if it is a solution (after τ-application) to the problem of proving $\tau(\mathtt{s}(\{a\}, \mathtt{System}))$ using the ALP program $\langle P, A, C\rangle$, where P is a logic program, A a set of abducible predicates and C a set of integrity constraints, such that:*

- *P includes CR1-CR11 (Definition 7) as well as:*

$$\tau(\{a\} \sqsubseteq C \in \mathcal{C}) \leftarrow \mathtt{classAssertion}(\tau(C), \tau(a))$$
$$\tau(\{a\} \sqsubseteq \exists r.\{b\} \in \mathcal{C}) \leftarrow \mathtt{objectPropertyAssertion}(\tau(r), \tau(a), \tau(b))$$

- *$A = \{\mathtt{classAssertion}, \mathtt{objectPropertyAssertion}\}$ and*
- *C includes the following constraints:*

$$\mathit{false} \leftarrow \mathtt{classAssertion}(\tau(C), \tau(a)) \wedge \mathtt{fresh}(\tau(a)) \quad \text{(IC1)}$$

$$
\begin{aligned}
\mathit{true} \leftarrow\ & \texttt{objectPropertyAssertion}(\tau(r), \tau(a), \tau(b)) \wedge \qquad \text{(IC2)}\\
& \texttt{objectPropertyAssertion}(\tau(r), \tau(a'), \tau(b')) \wedge\\
& ((\neg\texttt{fresh}(\tau(a)) \wedge \neg\texttt{fresh}(\tau(b'))) \vee\\
& (\neg\texttt{fresh}(\tau(a)) \wedge \neg\texttt{fresh}(\tau(a')) \wedge \tau(b) \neq \tau(b')) \vee\\
& (\neg\texttt{fresh}(\tau(b)) \wedge \neg\texttt{fresh}(\tau(b')) \wedge \tau(a) \neq \tau(a')))\\
\tau(a) = \tau(c) \leftarrow\ & \texttt{objectPropertyAssertion}(\tau(r), \tau(a), \tau(b)) \wedge \qquad \text{(IC3)}\\
& \texttt{objectPropertyAssertion}(\tau(r), \tau(a'), \tau(b)) \wedge\\
& \texttt{fresh}(\tau(a)) \wedge \neg\texttt{bound}(\tau(a'))\\
\tau(b) = \tau(d) \leftarrow\ & \texttt{objectPropertyAssertion}(\tau(r), \tau(a), \tau(b)) \wedge \qquad \text{(IC4)}\\
& \texttt{objectPropertyAssertion}(\tau(r), \tau(a), \tau(b')) \wedge\\
& \texttt{fresh}(\tau(b)) \wedge \neg\texttt{bound}(\tau(b'))
\end{aligned}
$$

Hypotheses generated during abduction are either class and role assertions from $\mathcal{C}$ (ground formulas) or formulas whose unbound variables are substituted with randomly generated terms. We assume the existence of a unary built-in predicate we denote `fresh`, whose term unifies with randomly generated UUID URIs [18].

Definition 2 includes two restrictions compared to the general problem of abduction. First, hypotheses should be of the form of role assertions. In Proposition 1, this restriction translates into the integrity constraint IC1. Second, at least two individuals in the resulting role assertions should not be fresh individuals, which we expressed as IC2.

Constraints IC3 and IC4 guarantee that abduction terminates, by limiting the number of fresh individuals that can be introduced. The number of possible role assertions that meet these constraints is polynomial with respect to the size of the original ABox. It follows that the overall discovery terminates in polynomial time, given that $\mathcal{EL}^{++}$ classification also terminates in polynomial time.

It is straightforward to prove that any solution produced by the ALP program of Proposition 1 is also a solution to the problem of WoT semantic discovery (up to the renaming of fresh individuals). However, to prove the converse, one should first observe that for any two CBoxes $\mathcal{C}, \mathcal{C}'$, such that individuals of $\mathcal{C}'$ can be substituted to obtain $\mathcal{C}$, it holds that if $\mathcal{C}' \not\models \alpha$, then $\mathcal{C} \not\models \alpha$. We do not develop a formal proof in this paper, left as future work.

5 Experiments

5.1 Experimental Setup

We applied the discovery framework presented in this paper on an experimental industrial workstation. This workstation 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; it had initally been designed as a water management plant model for educational purposes. The

original model has been added six micro-controllers, wired to the automation devices and acting as WoT servients with IP connectivity (over Wi-Fi). An overview of the workstation is provided in Fig. 1.

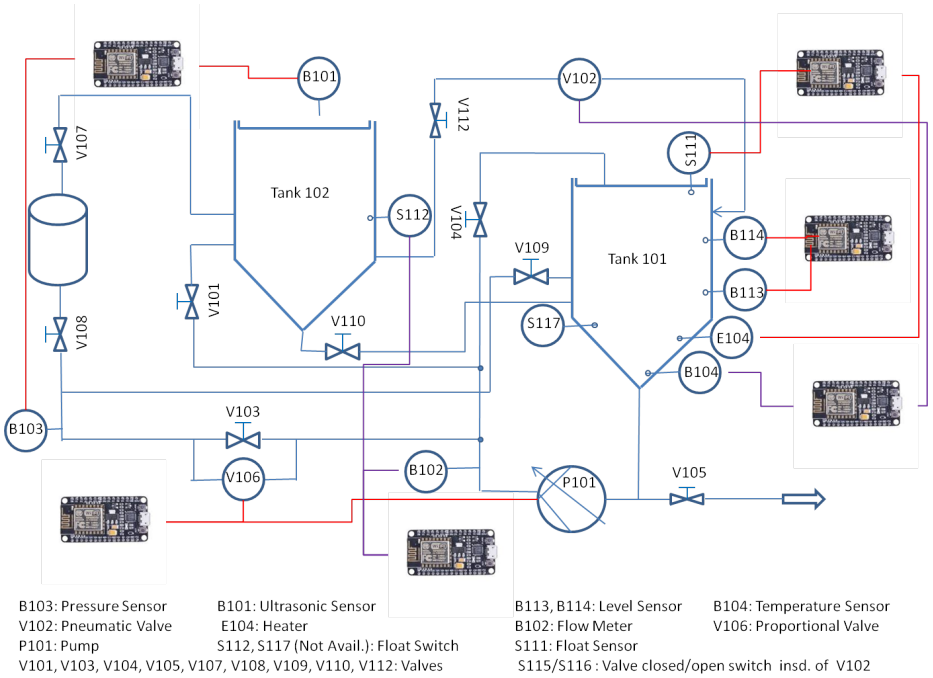


Fig. 1: Water management workstation with NodeMCU micro-controllers

These micro-controllers are ESP8266 (64kB RAM, 80 MHz), they all embed a lightweight RDF store designed for constrained devices (μRDF store [4]), and they can exchange RDF data encoded in binary JSON over CoAP, the Constrained Application Protocol. We recently showed that a combination of a JSON-LD compacted representation of RDF triples and EXI4JSON 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 [5]. The JSON-LD context we generated for this experiment includes all terms from the SOSA, SSN and TD ontologies, as well as a subset of eCl@ss concepts. Every ESP8266 servient stores a TD in its μRDF store. It can serve its TD and receive updates after discovery via CoAP `GET`, `PUT` and `DELETE` operations.

We implemented the ALP program of Definition 1 in Prolog, using the HYPROLOG framework [6]. Our program runs on a non-constrained machine, which acts as a Thing Directory used for discovery. It imports TD documents and knowledge bases with the `thea2` Prolog module [25]. Knowledge bases are

turned into normal form as described by Baader et al. [2] and TDs are also normalized—in the sense of RDF graph normalization—prior to Prolog import, with the Java tool `blabel`[13]. By convention, semantic resources in TDs are represented as RDF blank nodes, which are then replaced by canonical URIs after RDF normalization.

5.2 Discovery Task

Our experiment consists in building a graph of interaction for the six TDs exposed on the workstation. These TDs are defined against SOSA, SSN, TD and eCl@ss. The CBox we perform reasoning on includes the axioms of these four ontologies, as well as the definition of five systems that can be discovered by combining servient capabilities. These systems, whose definitions we manually provide, 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.

All these systems are rather simple and were designed to study the feasibility of our approach only. We do not address performance and scalability issues. Each system is meant to be the combination of two sub-systems, hosted by one or more servients. One can note that these systems themselves can be combined to perform more elaborate (and more realistic) tasks. Our semantic discovery approach would still cover these cases. As mentioned at the end of Sec. 3, the result of semantic discovery depends on the axioms we define. For our experiments, we defined axioms on systems such that all systems that can be discovered are associated to Web resources, as in the following example.

[13] http://blabel.github.io/

Example 3. In our experiments, the full axiom defining `ValveControlSystem` is as follows (note the use of the existential construct ∃`href.WebResource`):

$$\begin{aligned}
&\exists \texttt{hasSubSystem}.(27292200 \sqcap \\
&\quad \exists \texttt{madeActuation}.(\exists \texttt{usedProcedure}.(\exists \texttt{href.WebResource})) \sqcap \\
&\exists \texttt{hasSubSystem}.(27273201 \sqcap \\
&\quad \exists \texttt{isHostedBy}.27273201 \sqcap \\
&\quad \exists \texttt{madeObservation}.(\exists \texttt{usedProcedure}.(\exists \texttt{href.WebResource})) \\
&\sqsubseteq \texttt{ValveControlSystem}
\end{aligned}$$

After applying our Prolog program on the six TDs (ABox) and the CBox described above, we obtain height edges in the graph of interactions (computation time below 1s). All five 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. 2. 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). Here, one of the nodes has a degree of seven, for a maximum degree of sixteen. If the servient is decommissioned and removed from the network, half of the discovered systems would stop functioning.

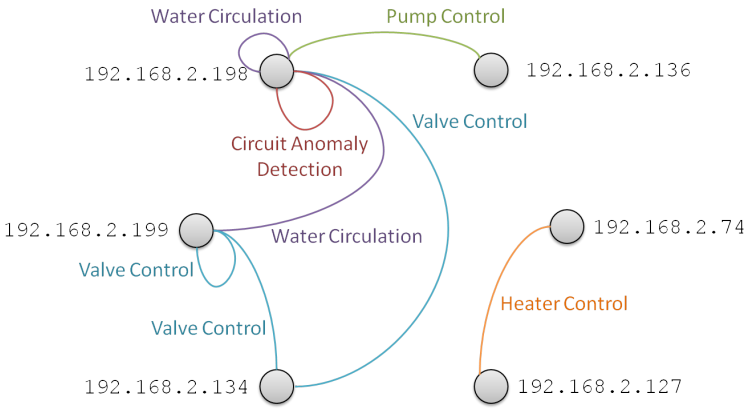
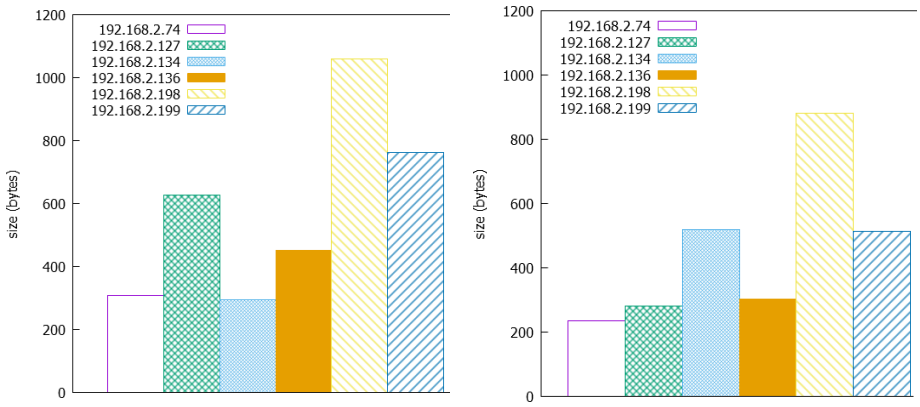


Fig. 2: Interaction graph resulting from semantic discovery on a water management workstation (servients are identified by their IP address)

In addition, we looked at the exchange that takes place between the Thing Directory 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 `POST` requests sent by individual servients to the Thing Directory or by a `GET` request broadcast by the Thing Directory to all μRDF store instances on the network. In the former case, TDs are put in the request payload while in



(a) Thing Descriptions (b) Results

Fig. 3: Size of knowledge bases exchanged between servients and Thing Directory, encoded in EXI4JSON (servients are identified by their IP address)

the latter case, they are in the response payload. In Fig. 3a, we show the size of the payload each servient sends.

In a peer-to-peer scenario, in addition to gathering servients' TDs, the Thing Directory must update them, as per Definition 4. Updates are performed as `PUT` requests sent by the Thing Directory to each μRDF store, with statements to add in the request payload. Payload sizes for updates are shown in Fig. 3b. 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 max. size 1024 bytes), which represents no technical challenge for micro-controllers like the ESP8266.

All RDF files and results shown in this paper are available online[14].

6 Conclusion

The problem we addressed in this paper is that of semantic discovery in WoT. We formalized it as an abductive reasoning problem over knowledge bases that use standard ontologies, among others the SOSA/SSN and TD ontologies. The former provides semantics for systems and features of interest, which are, in a generic fashion, referred to as 'things' in WoT.

Our logic programming approach and its implementation over a water management workstation with six servients show that constrained devices like microcontrollers can carry enough knowledge in the form of TD documents, such that meaningful reasoning can be performed to make spontaneous systems form in an autonomous fashion. Moreover, even though discovery is centralized, having

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

peer-to-peer systems, robust to changes, is still possible if the knowledge base carried by individual servients can be updated with the results of discovery.

In other words, the framework we present in this paper supports the vision of WoT as a large scale multi-agent system as presented in introduction, where devices can be quickly deployed and autonomously perform various tasks. To fully realize it, large scale experiments addressing performance, scalability and practicability issues are yet to be made. One condition to be fulfilled before large scale tests are even possible is to make extensive expert-reviewed system descriptions available on the Semantic Web. As we modeled them, sytem descriptions can be shared with existing Semantic Web technologies as regular Web ontologies. However, scaling up the engineering of such ontologies remains a challenge.

References

1. SmartM2M; smart appliances reference ontology and oneM2M mapping, 2015.
2. Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the EL envelope. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI 2005)*, volume 5, pages 364–369, 2005.
3. 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*. CEUR-WS.org, 2016.
4. Victor Charpenay, Sebastian Kaebisch, and Harald Kosch. μRDF store: Towards extending the semantic web to embedded devices. In Eva Blomqvist, Katja Hose, Heiko Paulheim, Agnieszka Lawrynowicz, Fabio Ciravegna, and Olaf Hartig, editors, *The Semantic Web: ESWC 2017 Satellite Events*, volume 10577, pages 76–80. Springer International Publishing, 2017.
5. Victor Charpenay, Sebastian Kaebisch, and Harald Kosch. Towards a binary object notation for RDF. In *Proceedings of the 15th Extended Semantic Web Conference (ESWC)*. Springer, 2018.
6. Henning Christiansen and Veronica Dahl. HYPROLOG: A new logic programming language with assumptions and abduction. In Maurizio Gabbrielli and Gopal Gupta, editors, *Logic Programming*, pages 159–173. Springer Berlin Heidelberg, 2005.
7. Jianfeng Du, Kewen Wang, and Yi-Dong Shen. A tractable approach to ABox abduction over description logic ontologies. In *AAAI*, pages 1034–1040, 2014.
8. Dov M. Gabbay, C. Hogger, and J. A. Robinson. Handbook of logic in artificial intelligence and logic programming, 1998.
9. Marco Gavanelli, Evelina Lamma, Fabrizio Riguzzi, Elena Bellodi, Riccardo Zese, and Giuseppe Cota. Reasoning on datalog±ontologies with abductive logic programming. *Fundamenta Informaticae*, 159(1-2):65–93, 2018.
10. 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)*, 2009.
11. Armin Haller, Krzysztof Janowicz, Simon J D Cox, Danh Le Phuoc, Kerry Taylor, and Maxime Lefrançois. Semantic Sensor Network Ontology. W3C and OGC Recommendation, W3C & OGC, October 19 2017.

12. Armin Haller, Krzysztof Janowicz, Simon J.D. Cox, Maxime Lefrançois, Kerry Taylor, Danh Le Phuoc, Josh Lieberman, Raúl García-Castro, Rob Atkinson, and Claus Stadler. The modular SSN ontology: A joint W3C and OGC standard specifying the semantics of sensors, observations, sampling, and actuation. *Semantic Web Journal*, 2018.
13. Krzysztof Janowicz, Armin Haller, Simon J.D. Cox, Danh Le Phuoc, and Maxime Lefrançois. SOSA: A lightweight ontology for sensors, observations, samples, and actuators. *Journal of Web Semantics*, 2018.
14. Sebastian Kaebisch and Takuki Kamiya. Web of things (WoT) thing description, 2017.
15. A. C. Kakas, R. A. Kowalski, and F. Toni. Abductive logic programming. 2:719–770, 1992.
16. Kajimoto Kazuo, Matthias Kovatsch, and Uday Davuluru. Web of things (WoT) architecture, 2017.
17. Szymon Klarman, Ulle Endriss, and Stefan Schlobach. ABox abduction in the description logic $\mathcal{ALC}$. 46:43–80, 2010.
18. P. Leach, M. Mealling, and R. Salz. A universally unique IDentifier (UUID) URN namespace, 2005.
19. Maxime Lefrancois. Planned ETSI SAREF extensions based on the W3C&OGC SOSA/SSN-compatible SEAS ontology patterns. CEUR-WS.org, 2017.
20. João Moreira, Laura Daniele, Luis Ferreira Pires, Marten van Sinderen, Katarzyna Wasielewska, Pawel Szmeja, Wieslaw Pawlowski, Maria Ganzha, and Marcin Paprzycki. Towards IoT platforms integration: Semantic translations between W3C SSN and ETSI SAREF. CEUR-WS.org, 2017.
21. Stefan Poslad. *Ubiquitous Computing: Smart Devices, Environments and Interactions*. John Wiley & Sons, Inc., 2009.
22. Sebastian Rudolph. Foundations of description logics. In Axel Polleres, Claudia d'Amato, Marcelo Arenas, Siegfried Handschuh, Paula Kroner, Sascha Ossowski, and Peter Patel-Schneider, editors, *Reasoning Web. Semantic Technologies for the Web of Data*, pages 76–136. Springer Berlin Heidelberg, 2011.
23. Stuart J. Russell. *Artificial intelligence: a modern approach*. Prentice Hall series in artificial intelligence. Prentice Hall, 3rd ed edition, 2010.
24. Ioan Toma, Elena Simperl, and Graham Hench. A joint roadmap for semantic technologies and the internet of things. In *Proceedings of the Third STI Roadmapping Workshop, Crete, Greece*, volume 1, pages 140–53, 2009.
25. Vangelis Vassiliadis, Jan Wielemaker, and Chris Mungall. Processing OWL2 ontologies using thea: An application of logic programming. In *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*. CEUR-WS.org, 2009.