Semantic based middleware to support nomadic users in IoT-enabled smart environments

Benoit Christophe

THÈSE DE DOCTORAT DE
l'UNIVERSITÉ PIERRE ET MARIE CURIE

Spécialité

Informatique

École doctorale Informatique, Télécommunications et Électronique (Paris)

Présentée par

Benoit CHRISTOPHE

Pour obtenir le grade de
DOCTEUR de l'UNIVERSITÉ PIERRE ET MARIE CURIE

Sujet de la thèse :

Semantic based middleware to support nomadic users in IoT-enabled smart environments

soutenue le 07 Septembre 2015

devant le jury composé de :

M. Jerome EUZENAT	Rapporteur
M. Serge CHAUMETTE	Rapporteur
M. Animesh PATHAK	Examinateur
M. Christophe MARSALA	Examinateur
M. Bertrand GRANADO	Directeur de l'EDITE

A ma famille, sans qui rien n'aurait été possible.

The above proposition is occasionally useful.
Bertrand Russell, about 1+1=2
Principia Mathematica

Acknowledgments

It would not have been possible to write this doctoral thesis without the help and support of the kind people around me, to only some of whom it is possible to give particular mention here.

Above all, I would like to thank my wife Manami for her personal support and great patience at all times. I am also grateful to my two wonderful daughters Noa & Lia that gave me the force and courage to finalize this thesis, through their joy of life. My parents have given me their unequivocal support throughout, as always, for which my mere expression of thanks likewise does not suffice.

Amongst my fellows at Bell Labs, I am deeply grateful to Matthieu Boussard that worked together with me over the last 7 years, always triggering the good pointers allowing to push my research further. I would particularly thank Alonso Silva, Vincent Verdot and Vincent Toubiana for their kind advices and the moments that we spent together at Bell Labs, exchanging on innovative ideas.

I address warm thanks to my former colleagues Philippe, Gerard, Olivier, Nicolas, Monique, Eric, Lionel, Dohi, Cedric and Pierrick.

I am also grateful to Martin Vigoureux for having made possible this research and the resulting doctoral thesis. I thank my colleagues Wenyi, Bela, Dominique, Francois, Laurent, Ludovic, Michel, Nicolas, Pierre, Richard and Thai.

Finally, I would like to acknowledge the financial support provided by Alcatel-Lucent and Bell Labs for writing this thesis.

For any errors or inadequacies that may remain in this work, of course, the responsibility is entirely my own.

Abstract

With the growth in Internet of Things, the realization of environments composed of diverse connected resources (devices, sensors, services, data, etc.) becomes a tangible reality. Together with the preponderant place that smartphones take in the daily life of users, these nascent smart spaces pave the way to the development of novel types of applications; carried by the phones of nomadic users and dynamically reconfiguring themselves to make use of such appropriate connected resources. Creating these applications however goes hand-in-hand with the design of tools supporting the nomadic users roaming in these spaces, in particular by enabling the efficient selection of resources. While such a selection calls for the design of theoretically grounded descriptions, it should also consider the profile and preferences of the users. Finally, the rise of (possibly mobile) connected resources calls for designing a scalable process underlying this selection.

Progress in the field is however sluggish especially because of the ignorance of the stakeholders (and the interactions between them) composing this eco-system of "IoT-enabled smart environments". Thus, the multiplicity of diverse connected resources entails interoperability and scalability problems. While the Semantic Web helped in solving the interoperability issue, it however emphasizes the scalability one. Thus, misreading of the ecosystem led to producing models partially covering connected resource characteristics.

Revolving from our research works performed over the last 6 years, this dissertation identifies the interactions between the stakeholders of the nascent ecosystem to further propose formal representations. The dissertation further designs a framework providing search capabilities to support the selection of connected resources through a semantic analysis. In particular, the framework relies on a distributed architecture that we design in order to manage scalability issues.

The framework is embodied in a VR Gateway further deployed in a set of interconnected smart places and that has been assessed by several experimentations.

Keywords

Smart environments, Internet of Things, Ubiquitous Computing, Pervasive Computing, Semantic Web, Knowledge Representation, Similarity measure, Distributed Information.

Résumé

Avec le développement de l'Internet des Objets, la réalisation d'environnements composés de diverses ressources connectées (objets, capteurs, services, données, etc.) devient une réalite tangible. De plus, la place prépondérante que les smartphones prennent dans notre vie (l'utilisateur étant toujours connecté) font que ces espaces dits 'intelligents' ouvrent la voie au développement de nouveaux types d'applications; embarquées dans les téléphones d'utilisateurs nomades – passant d'un environnement connecté (la maison) à un autre (la salle de réunion) – et se reconfigurant dynamiquement pour utiliser les ressources de l'environnement connecté dans lequel celles-ci se trouvent. La création de telles applications va cependant de pair avec le design d'outils supportant les utilisateurs en mobilité, en particulier afin de réaliser la sélection la plus efficace possible des ressources de l'environnement dans lequel l'utilisateur se trouve. Tandis qu'une telle sélection requiert la définition de modèles permettant de décrire de façon précise les caractéristiques de ces ressources, elle doit également prendre en compte les profils et préférences utilisateurs. Enfin, l'augmentation du nombre de ressources connectées, potentiellement mobiles, requiert également le développement de processus de sélection qui "passent à l'échelle".
Des avancées dans ce champ de recherche restent encore à faire, notamment à cause d'une connaissance assez floue concernant les acteurs (ainsi que leurs interactions) définissant (i.e., prenant part à) l'éco-système qu'est un "espace intelligent". En outre, la multiplicité de diverses ressources connectées implique des problèmes d'interopérabilité et de scalabilité qu'il est nécessaire d'adresser. Si le Web Sémantique apporte une réponse à des problèmes d'interopérabilité, il en soulève d'autres liés au passage à l'échelle. Enfin, si des modèles représentant des "espaces intelligents" ont été développé, leur formalisme ne couvre que partiellement toutes les caractéristiques des ressoures connectées. En particulier, ces modèles tendent à omettre les caractéristiques temporelles, spatiales où encore d'appartenance liées à l'éco-système dans lequel se trouvent ces ressources. S'appuyant sur

mes recherches conduites au sein des Bell Labs, cette dissertation identifie les interactions entre les différents acteurs de cet éco-système et propose des représentations formelles, basées sur une sémantique, permettant de décrire ces acteurs. Cette dissertation propose également des procédures de recherche, permettant à l'utilisateur (ou ses applications) de trouver des ressources connectées en se basant sur l'analyse de leur description sémantique. En particulier, ces procédures s'appuient sur une architecture distribuée, également décrite dans cette dissertation, afin de permettre un passage à l'échelle.

Ces aides à l'utilisateur sont implémentées au travers de briques intergicielles déployées dans différentes pièces d'un bâtiment, permettant de conduire des expérimentations afin de s'assurer de la validité de l'approche employée.

Contents

Introduction

Since its introduction by Weiser in 1991[Wei91], ubiquitous computing has been a fertile ground for research and technology, leading to a number of concrete advances, e.g. in mobile computing. However the core of the vision, revolving around smart spaces in which mobile users seamlessly consume services and information, has still a long way to go. Until recently, this was explained by a number of factors, two of them being without a doubt the need for pervasive network connections and the cost of embedding advanced electronics in everyday objects. While the latter has decreased dramatically in the past few years, the former has benefited both from the Internet of Things research field and from technology improvements, which make the Internet today a commodity available largely, and accessible to most connected devices.

A consequence of this commodity results in an increasing set of network-capable devices becoming part of people' daily life. As an instance and according to the International Telecommunication Union (ITU) [1], cell phone subscriptions are about 6.8 billions globally, with a total number of 7.1 billions people on the planet (96% penetration). Information Handling Services (IHS) [2] company reported that in 2013 the installed base and worldwide new shipments of Internet connected devices was a market of approximately 12 billions devices, with an expected growth of 8 billions on the horizon 2015, suggesting an exponential increase. This growing trend in providing smart components as well as connected sensors and actuators, is consequently leading to billions of services and data offered in a plurality of diverse smart environments (found in homes, factories, malls, etc.) through different and heterogeneous IoT devices. In such a context, realizing Mark Weiser's vision of ubiquitous computing now accounts for providing efficient tools and interfaces that will support users. In particular, search capabilities assisting users to find and further use

1. http://www.itu.int/en/ITU-D/Statistics/Documents/statistics/2013/ITU_Key_2005-2013_ICT_data.xls

2. Internet Connected Devices: Evolving from the "Internet of Things" to the "Internet of Everything", http://www.ihs.com/info/sc/a/internet-of-things.aspx

meaningful devices, are de-facto pre-requisites that need to be carefully designed according to the IoT specificities, e.g., where devices evolve over the time: may be relocated, may become unavailable, etc.

The preponderant place that smartphones take in the daily life of users further adds a mobility challenge to provide such tools, considering the continuity of service that IoT-based applications carried by these smartphones may need to ensure whatever the smart environment they are executed in. More specifically, realizing the vision of Mark Weiser also accounts for proposing mechanisms allowing the IoT applications of the nomadic users to be dynamically reconfigured with appropriate connected devices each time such users roam across a different smart environment.

Accordingly, this dissertation identifies the underlying challenges associated to the design of search capabilities for these IoT-enabled smart environments. Grounded by the ideas of Kindberg et al. [KBM+02] – that considered the realization of smart environments through the Web presences of nomadic people, places and things – this dissertation focuses on enabling mobile users to consume composite services and possibly aggregated information through connected devices available in the different smart environments that they cross as well as in the Cloud. Specifically, this dissertation is organized around three research axes that have been investigating over the last six years. The first axis lays the foundations for enabling user-personalized search capabilities by determining appropriate representations to formally describe devices and the requirements arising from applications. This axis also points out works that I have been launching then supervising, and consisting of defining user profiles by use of Fuzzy logic. The second axis investigates how accurate results can be returned to a user having performed a query, focusing on the semantics underlying the representation models. In particular, it yields to the design of a method that independently of any context, computes accurate similarity results between any entities having been described with such a semantic. Finally, the third axis focuses on creating a distributed infrastructure composed of cooperating smart environments to ensure a responsive and still relevant search process to any user. We motivate this work based on the emerging eco-system and the underlying problems that are becoming increasingly relevant to address.

Emerging eco-system

In our vision, the development of the Internet of Things has led to an eco-system in which five different stakeholders may be involved in scenarios where users consume services and information when roaming in smart environments.

Represented by Figure 1, this eco-system is composed of **Connected Devices** – comprising sensors, actuators and other smart components. Some of these devices can be mobile (e.g., a car) but are always associated with a location. Some can have availability restrictions (e.g., aligned with office hours), access policies as well as restrictions in terms of concurrency of access. Thus, the increasing number of connected devices accounts for a high heterogeneity in terms of the functionalities, services, etc. that they can offer.

A second actor that this eco-system implies is the **Applications**. Sharing similar characteristics with the first actor, they can be mobile (e.g., embedded in the smartphone of the user), with a use restricted to an environment (e.g., within the walls of a company) or can be living in the **Cloud**.

The third actor of this system is the mobile **User**. Coming with specificities (e.g., having a handicap, being in a hurry), a user has a profile defining his preferences and is often associated to a context.

The fourth actor of the eco-system is the **Smart Environment** that represents a place (a mall, a house, a room, etc.) equipped with an infrastructure that (at least) allows accessing to Connected Devices and Applications.

Finally, the last actor of the eco-system is the **Cloud** enabling a pervasive access (i.e., an access unbound to a geographical area) to Applications and Connected Devices and taking more and more importance with the rise of platforms such that Xively [3]).

In terms of interlinks, Connected Devices and Applications share a lot of commonalities. Both can be **localized** and only accessed at some specific location(s) (a house, a company, a mix of several locations, etc.) or can be **exposed** in the Cloud to be accessed independently from where they are. Both can also be **carried** by people (e.g., sensors in their smartphone). Most of the time both are **owned** by one – or a group of – person, leading to setting and managing different **access policies**. Finally, both are **consumed** by Users.

Accessing to a Connected Device, from a user viewpoint, may depend on the type of interaction that is expected. In particular, two cases can be distinguished. A *direct access*

3. Xively – Public Cloud for the Internet of Things, `https://xively.com/`

may be required for physical interactions (e.g., physically pushing a button on a device) while an indirect access (through the Internet) may be sufficient for other purposes (e.g., reading of a sensor value). Finally, Connected Devices are also going to be **integrated** in Applications. In particular, Connected Devices may be **coupled** with other devices in order to provide a particular service (coupling a lamp and a phone to provide a new type of call notification) or simply aggregated to deliver a feed of information (e.g., aggregating several data produced by the sensors of a house, for monitoring purpose). Obviously, using Connected Devices will depend on User profile and expectations as well as on Application requirements.

Other interlinks that form this eco-system concern the (potential) mobility of Users, Applications and Connected Devices accounting for the evolving context associated to any smart environment, i.e., where Users and their associations with Applications and Connected Devices change over time. Finally smart environments may also be interconnected, for instance in the case where they are part of the same indoor environment (e.g., a shopping mall where a smart clothes shop is interconnected with a smart coffee shop). Amongst others, such interconnections may enable different smart environments to share knowledge about their own context such that the associations between Users, Applications and Connected Devices mentioned before. Altogether, these different stakeholders and their interlinks allow for the creation of unprecedented data and services and create a kind of collective intelligence for supporting multiple scenarios. Let us consider one representative situation encountered in this eco-system.

Motivating application: Roaming across smart environments

Ben is visiting his friend Vincent. As he does not want to miss the last metro of the evening, he opens his application center on his mobile and searches for an application that could help. He selects a “Metro Warning” application that describes itself as requiring a public transportation information feed as input and a signaling device in the immediate environment of the user as output to warn the user of an incoming train. The system helps Ben in the configuration process by filtering automatically spaces and objects of relevance according to his context: it proposes the nearby metro stations that expose relevant information (upcoming trains) and returns only local, signaling devices made accessible by Vincent for his visitors (e.g. a connected lamp or a screen). The application lets Ben set a timer that will expire 3 minutes before the train arrives in the station so he has enough

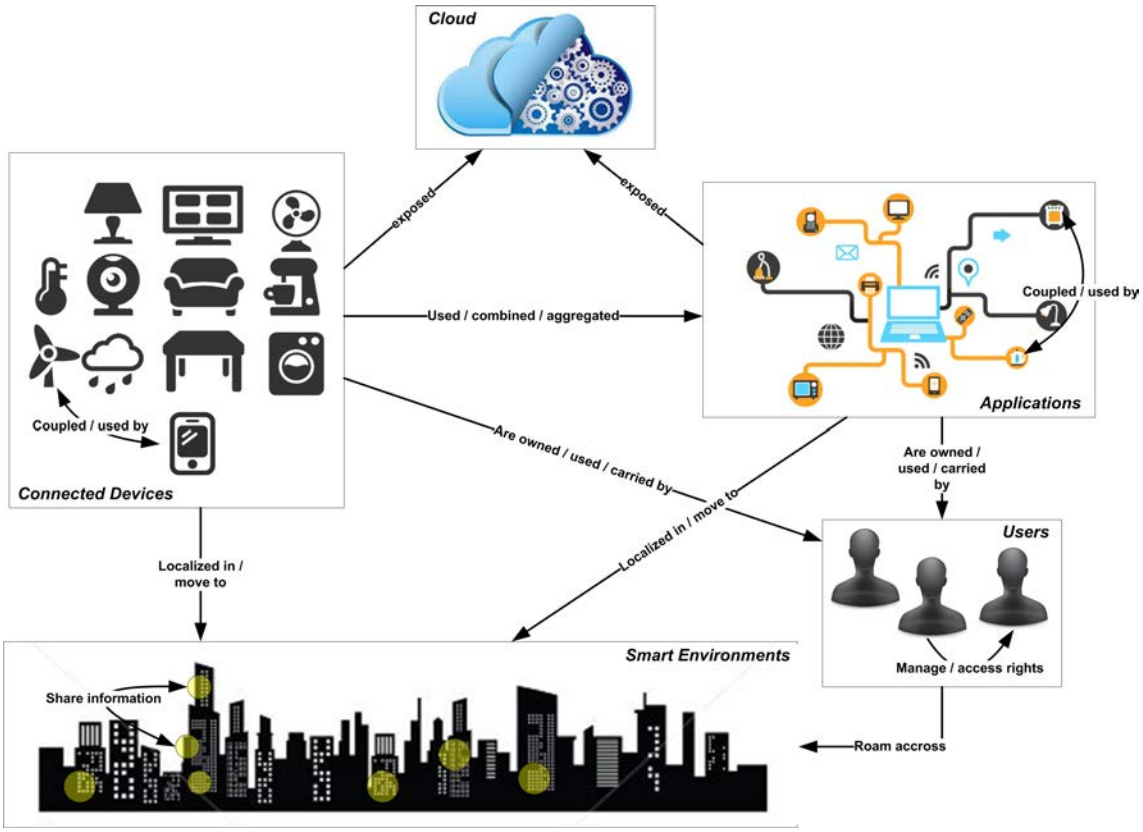


Figure 1 – The different stakeholders and their interconnections in the Internet of Things

time to catch it.

During the discussion, Ben decides to show his latest vacation picture to Vincent. He is already a user of a "Local Slideshow" application that he has set-up by coupling his home's Network Attached Storage (NAS) with his home's picture frame. As he resumes it, the application proposes to reconfigure itself. Some resources are kept unchanged and are remotely accessed (the NAS device) while locally available devices are proposed to supersede others – Ben selects amongst the proposed possibilities Vincent's video projector and the two discuss vacation plans. Later that evening, as the last train is 3 minutes away from the station, the "Metro Warning" application executes, issuing a visual signal on Vincent connected lamp. Ben packs his things and leaves. The "Local Slideshow" application is aware of Ben's departure and automatically stops the association with Vincent's connected video projector.

The day after, Ben reaches the new offices of his company and tries finding a nearby printer to get a paper version of the documents he has to show. He enters into the smart lobby equipped with a gateway in charge of managing devices and sends a request through his mobile phone to find such printer. Although no printer is localized in the lobby, the gateway returns one physically located in a nearby corridor. Indeed, after this printer was installed, its associated representation was sent to the gateway associated with the corridor as well as to all others in its vicinity (rooms, other corridors and the lobby).

After his presentation, Ben is willing to find a room equipped with a video conference system and hence, issues a request from his current location to the associated gateway (e.g., the gateway managing the meeting room where Ben gave his presentation). As the gateway is not aware of any video conference system in the different places it has received information from, he looked at Ben's profile to determine if it can spread the request to other gateways, located few steps ahead from the gateways it already knows. Ben's profile enables determining the maximum number of steps, allowing to fulfill the request in a set of places complying with Ben's expectations.

Difficulties in enabling the use of smart spaces

Despite multiple efforts to support mobile users in smart environments, realizing an application like the one mentioned here is still a hard and lengthy process for many reasons, as exposed hereafter.

Connected devices and applications are ill-defined

Practical problems (like the application discussed) require a combination of services from different resources able to solve user needs. This requires heterogeneous services, data and applicative requirements to be annotated with a common representation that is generic and does not need to be redefined every time a new resource must be taken into account. This calls for a representation capturing enough semantics and computational details so that it can support a variety of pervasive scenarios. In particular, the representation must enable the translation of an applicative requirement to a set of (possibly ordered) operations performed on data and services of different connected devices. In almost all scenarios, this representation will contain processable (homogeneous) temporal and spatial information e.g., to support the selection of available devices in a specific area. This representation may also contain information about access rights, to support the selection of accessible devices or applications. Finally, this representation may be able to describe the dynamics of a connected device or an application, e.g., how its access rights evolve over the time. All the aforementioned points are illustrated by the discussed application, as it needs to propose connected devices shared by their owner and located in the vicinity of the user. In addition, this application requires a method to couple the output coming from the NAS device with the input required by the video projector in order to instantiate the applicative requirement of visualizing remote pictures.

Lack of defining such a representation leads to an interoperability nightmare, limiting the development of so called IoT-based applications.

Users have specificities and preferences

While the selection of connected devices entails the definition of a homogeneous representation, it also requires taking care of user specificities especially when a direct access is required by the user. In particular, in the context of connected devices in smart indoor environments, the selection of connected devices may lead to different results whether the user is an elder or a youth or whether the user is in a good shape or not (e.g., having a handicap). Lack of a processable representation of the user profile may typically lead in proposing unsuitable devices to users.

Mobility entails reconfiguring applications

In most scenarios involving mobile users, applications must be able to dynamically reconfigure some of the resources they use. Indeed for a user in a given smart environment, the set of available resources tightly depends on their exposition. Smart environments may

expose some resources locally and therefore limit their use, while they could expose some others in the Cloud, allowing access from anywhere (through the Internet). In the context of applications carried by a user and interacting with local resources, a process capable of evaluating the similarities between different resources must be triggered each time users roam across different smart spaces, for reconfiguration purposes. Relying on the representation of the connected devices, this process should distinguish the preponderant features of interest from others as well as return exact or partial similarity results depending on the situation. While many tools can compute the similarity of different semantic representations, none of them has been customized to completely cover the semantics underlying the representation models of connected devices and applicative requirements.

Search processes must scale

Reconfiguring applications as well as proposing connected devices, requires scalable systems which can seamlessly handle huge volumes of data. In the context of billions of connected devices offered through the Internet, designing such systems accounts for characterizing the devices based on some criteria (one of them being their geo-location) to further create clusters of connected devices into which recommendations are given. Cluster based systems would allow to limit the search space to a few set of resources based on the characterization having been performed, enabling to propose reconfigurations in near real-time. Conversely, relying on a centralized approach where a plethora of connected devices have to be checked would avoid the development of applications as the one we discuss.

Contributions

This dissertation contributes to the problem of supporting mobile users roaming across different smart environments by detailing (resp. proposing) studies having been performed (supervised) over the last six years and directed towards providing search capabilities that can be applied in the addressed context. Overall these studies revolve around the following contributions:

1. To define a unified semantic representation for diverse connected devices and applicative requirements. This contribution is further completed by the proposition to scope the description of user profiles by introducing Fuzzy logic theory.
2. To design a process computing accurate semantic similarity measurements between

any entities described by the Description Logic (DL) $\mathcal{SHOIQ}$. Independent of any context, this study has been driven in order to design a similarity method dedicated to taking into account the semantics of this DL. Because the representations of connected devices and applicative requirements are covered by the semantics of this DL, the similarity measure can be applied as such to enable the selection of the most relevant connected devices for a given user' query.

3. To design a network composed of distributed clusters in which connected devices are gathered according to their location and that helps to reduce the search space by using the location as a filter.
4. To provide a framework where connected devices can be exposed, enabling the validation of the representation models by assessing the search capabilities having been designed.

Supporting mobile users roaming across smart environments accounts for providing search capabilities capable of returning meaningful results to users. In this dissertation a meaningful result – for a given user – is defined as ***a service or data offered by one (or a composition of) device(s), satisfying user needs in terms of the functionality or information that it delivers. Thus, this service or data is accessible, available, aligned with the specificities of the user and, depending on the scenario, is "localized" in the vicinity of the user***.

This definition emphasizes the different stakeholders participating in the "IoT-enabled smart environments" eco-system (displayed in Figure 1) and highlights the various key points that need to be addressed. Focusing on the definition of a unified semantic representation for connected devices and applicative requirements – considering their dynamics, the environment they are localized in, their access rights and temporal constraints – allows the development of specific searching procedures, supporting e.g., the dynamic (re)configuration of applications by matching their applicative requirements with connected devices. These procedures are undoubtedly depending on the preferences and expectations of users, which is allowed by relying on the formal model scoping their profiles.

Overall, determining the context tied to a search allows refining the space in which relevant devices must be found. In particular, this context is likely to depend on information about the user, its disabilities, its location, etc. With a refined search space, efficient methods relying on the semantic representation of connected devices and applicative requirements

can be developed to return meaningful devices.
While supporting the mobile user when roaming across different smart environments accounts for a wide range of scenarios, many of them may consist of returning connected devices in a specific geographical area e.g., for monitoring purposes, or in the vicinity of a given user e.g., constrained by physical interaction needs. In the case of a large smart environment, considering its representation to build a network of small and interconnected smart environments allows delineating search to a geographical area.
All aforementioned ideas are embodied in a framework that allows processing the formal descriptions of connected devices, applicative requirements and users. The framework also allows triggering the aforementioned search process, possibly applied on a specific geographical area.

Outline of the thesis

Chapter 1 recalls the basic concepts of the major technologies underlying this dissertation and as such, gives details about Description logic and Semantic Web technologies. This chapter further lays the definitions and notations used in the dissertation and in particular on the semantic similarity method that is detailed in Chapter 4. Chapter 2 surveys the most influential works having led to formal descriptions of connected devices and pervasive services and further describes the major studies in the field of semantic similarity computation. Chapter 3 defines a set of concepts capturing the semantics bound to connected devices and applicative requirements. This chapter also identifies the minimal Description logic that underlies these representations. Finally, this chapter details our viewpoint regarding the necessity to describe user profiles, the limitations of solely relying on Semantic Web technologies to create such profiles and finally how these limitations can be dissipated by the use of Fuzzy logic. In particular, the idea to rely on Fuzzy logic has been investigated under my supervision during a PhD thesis. Details of such work will therefore not be reported in this dissertation but some ideas can be found in [XMC13] and in the corresponding PhD dissertation [Xu15]. Chapter 4 presents our views regarding the design of efficient search procedures to return connected devices corresponding to a user' query. While these views include filtering devices using spatio-temporal as well as user profile information, this chapter does a strong emphasis on a method that performs similarity measurements by exploiting the specificities of the Description logic identified

in Chapter 3 [4]. Chapter 5 proposes the design of a federated network in which the problem of finding relevant connected devices is constricted by geographical boundaries, i.e., where the selected connected devices are necessarily nearby the user having performed the request. This chapter also provides the communication scheme used by the nodes of such a network. Chapter 6 presents a framework allowing to expose connected devices on the Web and in which the components detailed in the other chapters take place to enable search capabilities. Finally the last chapter discusses future challenges and concludes this dissertation.

4. Note that the proposed method is independent of the context in which it is applied. As a consequence, it may therefore serve to compute semantic similarity measurements in other contexts than the one associated to this dissertation, provided that the entities being compared are underpinned by the same DL

Chapter 1

Preliminaries

This chapter recalls the main notions that are used in this dissertation. In particular, it details the main logical constructs that are found in the various Description logics (DL) underlying Semantic Web technologies. Finally, a section of this chapter introduces the notations that will be used in this dissertation and in particular in the chapter presenting a novel semantic similarity measure for ontological concepts underlied by the DL $\mathcal{SHOIQ}$.

1.1 Background

1.1.1 Ontology and Description Logics

As defined by Gruber in 1993 [Gru93], an ontology is *an explicit specification of a conceptualization*, meaning that it represents concepts and objects that are presumed to exist in an area of interest. This declarative knowledge is underlied by Description Logics (DL), a representation language family particularly suited to define the concepts and properties of a domain of discourse [Hay79]. Based on a standard syntax and semantics, a DL specifies a canonical form to describe knowledge. In the literature, several DL with different expressive power have been proposed, based on studies evaluating the efficiency of their reasoning (e.g. their decidability and complexity) [DLNN91].
In its simplest expression, an ontology can consist of atomic concepts defined relatively to others by the sole use of the inclusion concept (e.g. assertions such as $A \sqsubseteq B$, $A \sqsubseteq C$ but not such as $A \sqsubseteq (B \sqcap C)$) [1]. In this dissertation, however, we primarily consider ontologies underlied by more expressive DL, such that $\mathcal{SHOIQ}$ (detailed in Table 1.1), involving the definition of complex concepts.

1. Note that in this case, the ontology is simply a hierarchy of atomic concepts

As usual, for C and D (possibly complex) concepts, $C \sqsubseteq D$ is called a General Concept Inclusion (GCI). A finite set of GCI forms the terminology $\mathcal{T}$ (or TBox) of the ontology. A role inclusion axiom is of the form $R \sqsubseteq S$ where R and S are two roles defined in the ontology. The set of all roles in an ontology is named N_R while we denote the set of all transitive roles with N_{R^+}. The finite set of role inclusion axioms represents a role hierarchy named $\mathcal{R}$. Note that we consider the terms *property* and *role* as equivalent, and that we use both throughout the paper.

In the case of ontologies underlied by the DL $\mathcal{SHOIQ}$, the pair $(\mathcal{T}, \mathcal{R})$ is called a knowledge base (KB). $\mathcal{T}$ represents the terminology of the ontology (the definitions of the concepts) and is also called the TBox. $\mathcal{R}$ denotes the set of roles defined in the ontology. Finally, the set of concept instances declared in the ontology is called the ABox and is sometimes written $\mathcal{A}$. For the same reason as [HS05] we do not include the ABox to the definition of the KB.

The formal semantics of $\mathcal{SHOIQ}$ is defined with respect to an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$. $\mathcal{I}$ consists of a non empty set $\Delta^{\mathcal{I}}$ called the domain of interpretation and an interpretation function $\cdot^{\mathcal{I}}$ mapping each concept C to a set of individuals $C^{\mathcal{I}}$ (such that $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and each role R to a set of pairs of individuals $R^{\mathcal{I}}$ (such that $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$).

With respect to $\mathcal{I}$, the DL $\mathcal{SHOIQ}$ allows the creation of atomic concepts (A such that $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$) and atomic roles ($R$ such that $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$). Complex concepts can be written using the operators *disjunction, conjunction, full existential qualification, qualified value restriction, qualified cardinal restriction* and *negation*. Concepts can also be *nominals*, i.e. defined as an enumeration of instances. Finally, roles can be organized hierarchically, can be transitive and finally can be inverse of other roles. Table 1.1 summarizes the semantics of the DL $\mathcal{SHOIQ}$. In this table, $x, y, z, a_1, \ldots, a_n$ denote individuals of $\Delta^{\mathcal{I}}$. n is a positive integer, while we use the symbol # to denote the cardinality of a set. Note that throughout the paper, we will use $\top$ to denote the absolute truth (as well as the *top* concept of the ontology, i.e., the concept that subsumes all the concepts defined in the ontology) and $\bot$ to denote the absurdity (as well as the *bottom* concept of an ontology, i.e. that is subsumed by all the concepts defined in the ontology).

The *least common subsumer* (LCS) of two concepts C and D is defined as in [CBH92] i.e. as the least common concept that subsumes both C and D. In such definition, the LCS results from a computation procedure and may lead to generating a new concept (with respect to what contains the ontology). The *least common ancestor* (LCA) denotes

Description	Syntax	Semantics
Atomic concept	A	$A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$
Atomic role	R	$R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$
Negation	$\neg C$	$\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
Conjunction	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Disjunction	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Nominal	$\{a_1, \ldots, a_n\}$	$\{a_1^{\mathcal{I}}, \ldots, a_n^{\mathcal{I}}\}$
Qualified value restriction	$\forall R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \forall y.(x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\}$
Full existential qualification	$\exists R.C$	$\{x \in \Delta^{\mathcal{I}} \mid \exists y.(x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\}$
Qualified cardinal restrictions	$\geq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \geq n\}$
	$\leq nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} \leq n\}$
	$= nR.C$	$\{x \in \Delta^{\mathcal{I}} \mid \#\{y \mid (x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\} = n\}$
Role Hierarchy	$R \sqsubseteq S$	$R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$
Role Inverse	$R \equiv S^-$	$R^{\mathcal{I}} \equiv \{(x,y) \mid (y,x) \in S^{\mathcal{I}}\}$
Transitive roles	$R \in N_{R^+}$	$\{(x,z) \mid (x,y) \in R^{\mathcal{I}} \wedge (y,z) \in R^{\mathcal{I}}\}$

Table 1.1 – Syntax and semantics of $\mathcal{SHOIQ}$

the most specific named concept that subsumes two concepts. Unlike the LCS, the LCA is obtained by solely using the subsumption graph of the ontology and never leads to computing a new concept.

Considering a set of axioms forming the terminology $\mathcal{T}$ of an ontology $\mathcal{O}$, we use the term *primitive concepts* to denote the set of concepts of $\mathcal{T}$ that occur only on the right-hand side of axioms. If concepts appear in the left-hand side of an axiom, then we use the term *defined concepts*. As an example, in the following axiom $C \equiv \forall R.A \sqcap B$, C is a defined concept, while A and B are primitive concepts (and R, a role).

1.1.2 The Semantic Web: OWL and SWRL

Over the last 15 years, a lot of effort has been done to realize the vision of a Semantic Web as defined by Tim Berners Lee [BLHL01], i.e., where Web resources are easily accessed and consumed in automated processes. Amongst others, the OIL language was developed by a group of (largely) European researchers in 2001 [FHHM01] while DAML-ONT was supported by the DARPA DAML program [MFSH03]. Merged to give DAML-OIL[MFHS02], further efforts were done at the instigation of the W3C to finally produce the Web Ontology Language (OWL [2]) and, on top of it, the Semantic Web Rule Language [3] (SWRL) coupling the use of OWL with the Unary/Binary Datalog RuleML

2. OWL 2 Web Ontology Language Document Overview (Second Edition), `http://www.w3.org/TR/owl2-overview/`

3. SWRL: A Semantic Web Rule Language Combining OWL and RuleML, `http://www.w3.org/Submission/SWRL/`

sublanguages of the Rule Markup Language in order to express complex concepts such that ones involving relations between individuals.

Overall, OWL can be described as a *computational logic-based language, such that knowledge expressed in OWL can be exploited by computer programs*[4]. More specifically, the concepts expressed in OWL rely on operators found in Description logic. The following briefly recalls what is SWRL (based on the words of the authors of the SWRL specification). Aligned with RuleML, SWRL rules are of the form of an implication between an antecedent (body of the rule) and consequent (head of the rule). According to the SWRL specification, both the antecedent (body) and consequent (head) consist of zero or more atoms. Multiple atoms are treated as a conjunction meaning that if an antecedent is composed of several atoms, all of them must be satisfied in order for the associated consequent to hold. Atoms in SWRL rules can be of the form $C(x)$, $P(x, y)$, sameAs(x, y) or differentFrom(x, y), where C is an OWL description, P is an OWL property, and x, y are either variables, OWL individuals or OWL data values. In SWRL, variables that appear both in the antecedent and the consequent of a rule are treated as universally quantified. SWRL variables that appear only in the antecedent of a rule are treated as existentially quantified. The scope of any SWRL variable is moreover limited to a given rule. As usual, only variables that occur in the antecedent of a rule may occur in the consequent (a condition usually referred to as "safety"). Aligned with OWL, SWRL supports the Open World Assumption, meaning that an inference is proved if and only if it can be proved that the contrary never holds. A consequence is that the inferences built by OWL and SWRL reasoning are monotonic (i.e., an inference cannot be invalidated by some others). Another consequence is that OWL and SWRL do not allow negation as a failure (as it could entail non-monotonic inferences). For the same reason, SWRL rules are not allowed to retract any information from an ontology.

Equation 1.1 gives an example of a SWRL rule. In this example, the rule is made of an antecedent of two atoms and a consequent of one atom. Verifying the conjunction of atoms of the antecedent implies that the atom in the consequent holds. Specifically in this example, if an OWL individual (say a) is in relation with another OWL individual (say b) through the property "hasFather". Then if this OWL individual b is itself in relation with a third OWL individual c through the property "hasBrother", then it holds that the

4. Web Ontology Language, OWL`http://www.w3.org/2001/sw/wiki/OWL`

individuals a and c are in relation with the property "hasUncle".

$$hasFather(?x, ?y) \wedge hasBrother(?y, ?z) \rightarrow hasUncle(?x, ?z) \tag{1.1}$$

1.2 Notations and Definitions

For a given ontology $\mathcal{O}$, we use the notation $\mathcal{W}_\mathcal{O}$ to define the set of all possible $\mathcal{SHOIQ}$ expressions that can be written by coupling the defined concepts and properties of $\mathcal{O}$ with the different $\mathcal{SHOIQ}$ operators. As a consequence, $\mathcal{T} \sqsubseteq \mathcal{W}_\mathcal{O}$. We use the term *pseudo-concepts* to denote the set of concepts that do not exist in the ontology but that are extracted by the method defined in Section 4.2 (i.e., these concepts have not been defined by an ontologist). The set of pseudo-concept is written $\mathcal{PS}$ and contains expressions formed by concepts and logical operators of the DL $\mathcal{SHOIQ}$. As a consequence, $\mathcal{PS} \sqsubseteq \mathcal{W}_\mathcal{O}$.

$\mathcal{G}_\mathcal{O}$ refers to the classification graph of $\mathcal{O}$ before it has been expanded by the method defined in Section 4.2, while $\mathcal{E}_\mathcal{O}$ accounts for the expanded classification graph. $\mathcal{G}_\mathcal{O}$ contains concepts of $\mathcal{T}$ while $\mathcal{E}_\mathcal{O}$ contains concepts of $\mathcal{T} \cup \mathcal{PS}$.

For the same reasons as [dFE06], eliciting the semantics of concepts in $\mathcal{T}$ requires to normalize them. Towards this goal, we define a $\mathcal{SHOIQ}$ Normal Form as follows [5].

Definition 1.1. A defined concept C_D is in $\mathcal{SHOIQ}$ Normal Form iff $C_D \equiv \bot$ then $C_D \coloneqq \bot$ or if $C_D \equiv \top$ then $C_D \coloneqq \top$ or if $C_D \equiv D_1 \sqcup \ldots \sqcup D_n (\forall i = 1 \ldots n, D_i \neq \bot)$ then each D_i is such that:

$$D_i \coloneqq \Big(\mathrm{Prim}(D_i)\Big) \bigcap \Big(\mathrm{Nom}(D_i)\Big) \bigcap \Big(\bigcap_{R \in N_R} \mathrm{Rest}(R, D_i)\Big)$$

with:

- Prim(C), the intersection of all (possibly negated) primitive concepts at the top-level of C,
- Nom(C), the intersection of all (possibly negated) nominals at the top-level of C,
- $\mathrm{Rest}(R, C) \equiv \mathrm{Exist}(R, C) \sqcap \mathrm{Univ}(R, C) \sqcap \mathrm{Atleast}(R, C) \sqcap \mathrm{Atmost}(R, C) \sqcap \mathrm{Exactly}(R, C)$ with:

5. Note that in our definition, the $\mathcal{SHOIQ}$ Normal Form neither involves hierarchy nor transitivity of roles and consequently could be seen as an $\mathcal{ALCOQ}$ Normal Form.

— $\text{Exist}(R, C) \equiv \bigcap_{C' \in \text{ex}(R,C)} \exists R.C'$, with $\text{ex}(R, C)$ being the set of all C' such that $\exists R.C'$ appears at the top-level of C (C' is a singleton in the case of $\exists R.a$ restrictions, with a, an individual).

— $\text{Univ}(R, C) \equiv \forall R.\text{val}(R, C)$, with $\text{val}(R, C)$ being the conjunction $C_1 \sqcap \ldots \sqcap C_n$ in the value restriction of role R (here again, C_i is a singleton in the case of $\forall R.a$ restrictions, with a being an individual).

— $\text{Atleast}(R, C) \equiv \bigcap_{C' \in \text{al}(R,C)} \geq n_{max} R.C'$, with $\text{al}(R, C)$ being the set of all C' such that $\geq nR.C'$ appears at the top-level of C and with n_{max} being the highest n if more than one minimal cardinality exist for the same C'.

— $\text{Atmost}(R, C) \equiv \bigcap_{C' \in \text{am}(R,C)} \leq n_{min} R.C'$, with $\text{am}(R, C)$ being the set of all C' such that $\leq nR.C'$ appears at the top-level of C and with n_{min} being the smallest n if more than one maximal cardinality exist for the same C'.

— $\text{Exactly}(R, C) \equiv \bigcap_{C' \in \text{exact}(R,C)} = nR.C'$, with $\text{exact}(R, C)$ being the set of all C' such that $= nR.C'$ appears at the top-level of C.

— Any sub-description C' in $\text{ex}(R, C)$, $\text{al}(R, C)$, $\text{am}(R, C)$, or $\text{exact}(R, C)$ and any $C_i \in \text{val}(R, C)$ is in Normal Form except if such C' has already been rewritten (allow handling cyclic definitions of concepts such that $C \equiv (\exists R.C)$).

Note that computing the $\mathcal{SHOIQ}$ Normal Form of a concept requires writing it in Disjunctive Normal Form (DNF). Such operation is always possible by pushing negation inwards, using De Morgan laws and the duality between "existential" and "universal" ($\neg\exists A \Leftrightarrow \forall\neg A$) as well as between "at most" and "at least" numbers restrictions.
As a convention the set of direct subsumers of any concept $C \in \mathcal{E}_\mathcal{O}$ will be written $\overline{S_D}(C)$ while its set of direct subsumees will be written $\underline{S_D}(C)$.

Chapter 2

Related Works

The proposed work lies at the intersection of multiple active research areas. Here we discuss the related works in the context of the problems studied. We first survey the related works which tackle the design of formal representations for connected devices and applicative requirements. In particular, we review the different approaches to context modelling with a focus on the case of ontologies and semantic middlewares having been developed to enable the use of pervasive environments. As searching in smart environments is often linked to determining the similarity between resources and because our approach is underlied by semantic models, we finally review the different semantic similarity measures having been developed over the years. We also position these measures according to one that is detailed in the Chapter 4 of this dissertation and that is based on concepts described in an ontology underlied by the Description Logic (DL) $\mathcal{SHOIQ}$.

2.1 Models for smart environments

Developing applications dedicated to smart environments is inherently complex as it requires to adapt to various changing context information, in particular those associated to users, connected devices and computational resources. These constantly evolving environments calls for designing adequate models and reasoning techniques in order to support the creation of evolvable context-aware applications [BBH+10]. Over the last ten years, the pervasive computing community has produced different modelling approaches directed in this goal, i.e., to formally describe connected devices, applications and environments. While these approaches were made to support some common requirements (e.g., to lower

services or data heterogeneity, to capture service mobility, to represent dependencies between services, to describe state, time and planning associated to a service, etc.), all of them differ in their expressiveness, the reasoning that they allow and the computation time which is required to process them.

Early models focused on representing context and were mostly dominated by key-value as well as markup scheme based approaches. The major weakness of these approaches, however, was their inability to derive higher level information e.g., using a reasoning procedure. As pointed out by [BBH+10], *the introduction of the W3C standard for description of mobile devices, Composite Capabilities / Preference Profile (CC/PP), saw the first context modelling approaches to include elementary constraints and relationships between context types.* Based on the Resource Description Framework (RDF) – a language designed by the W3C to represent any resources on the Web – CC/PP was inheriting a simple entailment regime [1] allowing to derive higher level context information. Limitations of RDF-based context modelling were however pointed out in various works [IRRH03, SLP04] and finally led to the design of more expressive approaches based on database or knowledge management techniques that the following subsections will detail.

2.1.1 Role-based approaches

Role-based approaches are mostly applied to context modelling and take their foundations from ORM (standing for Object Role Modelling) a method used to model the semantics of a universe of discourse. Mostly used in order to design conceptual models for information systems (in particular databases), ORM models are based on the paradigm of *objects that play roles* (e.g. a *Person* jogs, a *Person* is employed by a *Company* since an *Entry Date*, etc.). Specifically ORM describes relations between one or more object types through fact types, drawn as sequences of roles (accounting for representing n-ary relations). Each object type is assigned a name and a representation type (e.g., a type name can be *Person* and the representation can be an *Id* associated to a *Person*). Fact types are annotated with uniqueness constraints spanning over one or several roles, allowing to define restrictions. Represented through diagrams, object-role models are usually considered as close to what is expressed in natural language and are consequently assumed as easy to understand for developers and designers.

Based on ORM, the prominent work about context modelling was formalized by Henrick-

1. RDF Entailment Regime, `http://www.w3.org/TR/rdf-mt/#entail`

sen et al. through their Context Modelling Language (CML) [HIR02, HI04, HI06] in which they described extensions enabling to label fact types in order to let them represent types of context information. In particular, in CML a fact can be:

— static, e.g. as part of the basic description of an object type
— sensed, e.g. resulting from the output of a sensor
— derived, i.e. resulting from a deductive process
— profiled, e.g. resulting from user preferences

In addition, Henricksen et al. provided annotations allowing to label a fact as possibly transient (e.g., a role which is valid during a certain period of time, which records historical data or that predicts future data) as well capable of managing ambiguous information (e.g., describing conflicting location reports from a variety of location sensors).
Henricksen et al. also proposed to model situations through rules involving facts. In their approach, situations are defined as predicates of zero or more variables and checked against a set of facts defined in a CML model. Situations can further be coupled using logical connectives such as conjunction, disjunction, negation, universal and existential operators. Evaluation of the rules is either 'true', 'false' or 'possibly true' a state implied by their support for managing ambiguous information.

According to Henricksen et al., CML is much more adapted than the technologies underlain by Description Logics to model context information (in particular the Semantic Web technologies). Their argumentation relies on four major points: The possibility in CML to represent and reason over imperfect context data, the possibility in CML to represent temporal roles, the ability to represent n-ary relations in a 'natural language' fashion and the easiness to understand ORM compared to the Web Ontology Language (OWL) underlying Semantic Web approaches.
Such argumentation can however be discussed, especially the three first points. Indeed, in the Semantic Web roles can be annotated, these annotations being either a free text tag or a concept formally defined. Such annotations may comprehend temporal indications or quality of the information. As pointed in a note published by the W3C by Noy et al. n-ary relations can be emulated by using dedicated patterns when representing data. Such pattern can also be used in order to associate to a role additional attributes such as a probability that the role holds or some temporalities. It is however true that such patterns

complicate the understanding of the whole model. Finally additional SPARQL [2] queries can be written to emulate the management of uncertain data. As an instance, suppose that two sensors have located Ben. The first one is in the Kitchen, while the other is in the living room. While asking the system if 'Ben is in the Kitchen' would return 'possibly true' in a CML approach, it would return 'true' in a Semantic Web approach. To emulate a 'possibly true' answer, one would have to submit another query such that counting the locations where Ben is located and to count the number of results being returned. A result higher than '1' would entail that Ben is in the Kitchen (learnt from the first query) and somewhere else, leading to deducting an uncertain data and modifying the answer to something being 'possibly true'. Notwithstanding, one have to agree that such queries represent additional works for designers as the information is not encoded in the model.
One drawback of CML, however, lies in the flat structure entailed by the ORM method in which none of the object type (resp. role type) can be a sub/super-type of another. In this vision, it is for instance impossible to describe that a HD TV is a specialization of a TV, that a smart phone is a specialization of a phone, a camera and other connected devices. While this may not be important in some cases, we believe that it would limit the support of searching connected devices suitable for a given scenario. In particular, being able to compute similarity measurements would be quite limited if using this approach.
A second drawback of CML is implied by its expressiveness. In particular, its ability to define situation with disjunction and negation may suffer from computation time required to evaluate if some known facts match a rule.

2.1.2 Ontological models and semantic middlewares

Aside the role-based approach, the increasing popularity of ontologies at the beginning of the 2000's [BLHL01] led to the development of ontology-based models and semantic middleware solutions. In particular, such models and solutions gained momentum and became an important area of research [NAA09], with an enthusiasm accentuated by the advances performed in the Semantic Web community. Indeed, the development of (computationally) tractable Semantic Web profiles – OWL-DL – together with the realization of efficient reasoning tools paved the way to developing rich and processable context models i.e., where higher level context information could be derived.
Over the years, a plethora of models and associated middlewares have been designed,

2. SPARQL Query Language for RDF`http://www.w3.org/TR/rdf-sparql-query/`

all of them focusing on a particular domain (healthcare [LLD, PG11] , tourism [Knu04, DD09], transport [NK09, BGM+10, BVAC13], building automation [SVVB12, DP14], home [CNS+06, BC08, WH11, CKY+11], agriculture [CPLM11, WWG13], etc.) in order to better automate user's daily activities and tasks. Many studies focused on deriving context information [Che03, GS04, SLPF03, RMCM03, PBW+04, CGZK04, SPL06] as well as providing frameworks to enable the realization of composite and reconfigurable services, e.g. through agents or applications [KKK+08, MPG+08, ILMF11, RNS+08, SCM10]. The rise of the development of senseable objects resulted in multiple efforts to build models enabling to recognize a situation [CNS+06, KHF+11, CNW12, MDEK13]. Other models [ELES06, HWG07, Goo, CNBC10] as well as those surveyed in [CHN+09] were also dedicated to providing generic descriptions for sensors and the entities that they measure.
All these approaches share the point that defining a common semantics for connected devices, sensors, tasks, etc. enables to leverage their interoperability, stepping toward the vision expressed by [Las05] in which the Semantic Web technologies are sensed as being particularly well-suited to solve the *"interoperability nightmare"* introduced by ubiquitous computing.
The following subsections will present a large overview of the existing models and middlewares that we consider as relevant in the scope of this dissertation. In particular, we will concentrate on the models and middlewares enabling to derive context information and device discovery. We will however leave aside vertical models, too specific to a particular domain and consequently overlooking many aspects that this dissertation intends to cover.

Models to derive context information

Chen et al. [Che03] coupled the use of ontologies with agents, interconnected using the FIPA [3] specifications in order to develop a middleware allowing heterogeneous devices to cooperate. Specifically, Chen et al. used context-aware agents together with an autonomous broker (CoBrA, see [Che03]), to collect and process information about a given context (e.g. a weekly meeting in a given room). Computation of such context relied on the use of a Standard Ontology for Ubiquitous and Pervasive Applications (SOUPA) [CFJ05], defining formally intelligent agents (their actions, events, time & space constraints, etc.). Chen et al. also modelled users' privacy policies enabling the selection of connected devices

3. FIPA specifications represent a collection of standards which are intended to promote the interoperability of heterogeneous agents and the services that they can represent. `http://www.fipa.org/specifications/index.html`

according to users' rights. Note that the idea developed by Chen et al. was further reused in other studies, such that [BJH+04] which focused on sharing descriptions of context data to build context-aware architectures.

Gandon et al. [GS04] proposed the semantic e-Wallet concept to support automated identification of and access to personal resources (calendars, location functionalities, databases,etc.) in a context of multiple agents assisting a user to realize different tasks. Towards this goal, each resource was considered as a Semantic Web Service i.e., with semantically described functionalities. Additionally, the semantic e-Wallet let users create privacy policies (using a semantic model) enabling to prevent that agents access to restricted information.

Strang et al. [SLPF03] proposed an Aspect-Scale-Context (ASC) model represented through the CoOL ontology. In this model, an aspect is a feature of interest (Temperature, Pressure, etc.) while a scale is a metric (Meter, Celsius, Pascal, etc.). An aspect can be linked to several scales while a scale can be associated with several context information. The CoOL ontology has been specifically developed in order to support the evaluation of context queries, to detect context information inconsistencies.

Wang et al. designed the CONON [WZGP04] ontology for smart home environments and built the SOCAM [GPZ04] middleware. The idea of the authors was to avoid inconsistencies in objects' representation, enabling then their interoperability and the derivation of higher level context information.

The DAML+OIL ontology language has been the basis of the GAIA [RMCM03] middleware, introduced by Ranganathan et al. In GAIA, reasoning for deriving new context data was performed by means of rule-based inferences and statistical learning. Ontologies were used to provide a clear semantics to data derived through different reasoning techniques.

Preuveneers et al. [PBW+04] designed a context ontology in order to enable application adaptation, automatic code generation and mobility as well as generation of device specific user interfaces. Their approach focused on offering a definition for many context information in order to give as much processable information as possible to any application.

Christopoulou et al. [CGZK04] defined an ontology to enable the composition of different applications. In their approach, an object is modelled as an eGadget associated to a Plug representing functionalities offered by the object as well as its physical properties (e.g., shape). Composition of objects is modelled through a concept eGW that accounts for associating at least 2 functionalities of different objects. Orchestration, availability, location and applicative requirements are however out of the scope of this paper.

Singh et al. [SPL06] proposed a framework enabling the collaboration of different pervasive computing environments (called AS, standing for Autonomous Systems) by using Semantic Web technologies. Such ASes were semantically described, allowing then interconnections to be created. ASes were envisioned to be linked to places such as office, home or pub. Each AS runs a knowledge base maintaining the current context of a place (in terms of functionalities, privacy and trust), the resources and people it managed. The ASes came with a set of RDQL [4] rules allowing to take decision regarding what to do when a new resource joined the AS.

Situation recognition had also led to several middlewares participating to deriving higher level context information. Amongst other initiatives, Chen et al. [CNW12] described a middleware for continuous activity recognition based on multi-sensor data streams in smart homes. Underpinned by an ontology, their work consisted of modelling context information associated to smart homes as well as what they called "Activities Daily Living" (ADL), consisting of activities performed by users. Context information was represented by descriptions for sensors, actuators and other home entities, including time and location attributes. Activities were represented with a richer Description Logic, taking into account conditions, goals, effects and durations. Situation recognition was performed using semantic subsumption reasoning algorithms.

Kurtz et al. [KHF+11] proposed the OPPORTUNITY framework to recognize human activities with limited guarantees about placement, nature and availability of sensors. They divided their approach in three steps, including coordinating the recruitment of sensors according to a high level recognition goal and, the instantiation of data elements to infer activities. In their approach the coordination and recruitment of sensors is performed based on a semantic mapping between the descriptions of the goals and the descriptions of the sensors (capabilities, characteristics, etc.). Such descriptions are based on a common and formal model using OWL enabling to abstract and structure goals as well as functionalities of sensors. Such abstraction is important as it enables sensors to be superseded by others (when not available) or complex activities to be seen as a set of atomic ones.

More recently Mediskos et al. [MDEK13] proposed to define activity patterns in an ontology. Based on the Domain and Situations (DnS) pattern [GM03] described in the DOLCE+DnS Ultralite ontology [5], their patterns are further used in rules enabling to

4. RDQL, a Query Language for RDF, `http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/`
5. DOLCE+DnS Ultralite (DUL) ontology. `http://www.loa.istc.cnr.it/ontologies/DUL.owl`

recognize whether an activity takes place or not. In their representation, an activity contains participants and physical entities. An activity is also linked with location and time attributes. Two patterns have been developed, the first one enabling to determine if an activity is a specialization of a situation, and the second representing a situation as a sequence of activities. As relying on OWL, the authors are however unable to take into account the order in which activities must be performed to realize a situation.

Models to discover and compose devices

Aligned with Chen et al., Katasonov et al. [KKK+08] developed a middleware allowing heterogeneous devices to cooperate, based on the FIPA specifications. In particular, the authors relied on Semantic Web Service ideas [PL04] to create a Semantic Web of Things composed of agents presenting the semantic profiles of devices that they were monitoring. Their resulting infrastructure was processing incoming semantic requests by triggering appropriate device functionalities.

Song et al. [SCM10] proposed a middleware focused on task computing, i.e. enabling users to select different and heterogeneous devices to realize a task. They created an application layer to enhance interoperability and composability of services offered by these different devices, which consisted of wrapping the device semantics brought by existing specifications (such as Bluetooth, UPnP, etc.) into semantic services.

With Construct, Coyle et al. [CNS+06] proposed a semantic middleware to unify the data and service representation of diverse home automation systems. In Construct, each device in a smart home is considered as either a sensor or an actuator. Once semantically annotated, these devices are integrated into smart home applications.

Bonino et al. proposed DOGOnt, an ontology dedicated to classifying the appliances found in a smart home as well as the possible actions to perform on them [BC08].

With EASY, Ben Mokhtar et al. [MPG+08] associated semantics to devices of pervasive environments to further enable their discovery. They defined a Capability concept allowing service requesters to express their needs and service providers to advertise their capabilities. Besides, they defined conformance rules for matching services with requested properties. Finally, Ibrahim et al. [ILMF11] defined semantic equivalence relations between services to enable service substitution in pervasive environments.

The work done by Ruta et al. [RNS+08] consisted of defining an ubiquitous knowledge-based system to enable object discovery in smart environments. Smart environments are

considered as composed of micro devices, each of them annotated with semantic descriptions referring to concepts defined in ontologies. The description of a micro device provides intentional knowledge, i.e., characteristics specific to the micro device. Discovery of micro devices involves two steps. Upon the reception of a request, a pre-selection based on an ontology reference is made, to select a 'type' of resources. Thus, a matching is performed amongst semantic annotations of selected resources and an on-demand request.

Ostermaier et al. [ORM+10] proposed to search for *real-world entities having certain properties*. Their idea consisted of associating a Web page to a real-world entity (e.g. a meeting room) containing additional structured metadata about the sensors connected to it. The requests were made using a simple search language (e.g. 'room ABC occupancy:empty') and were composed of a 'static' part (e.g. 'room ABC') and a 'dynamic' part (e.g. 'occupancy:empty'). When a request was received by the search engine, the 'static' part was analyzed to filter the sensors providing a capability (e.g. filter the sensors of room ABC that provide 'occupancy'). The search engine was further using predictive models to compute the probability that filtered sensors return a given value (e.g. probability that sensors giving 'occupancy' in room ABC return 'empty'). When sufficient hits were found, the process was stopped and the results about a given property (occupancy) were returned to the user.

Pfisterer et al. [PRB+11] proposed an architecture aiming to open access to sensors, enhancing integration of data and services of heterogeneous sensors and facilitating novel applications. They provided vocabularies allowing to integrate descriptions of sensors and things with the Linked Open Data (LOD) cloud [6] and proposed search mechanisms taking into account states of sensors (e.g. availability). Their approach in answering a user request was to query a triple store by using the SPARQL protocol.

He et al. proposed the use of semantics to define the *context of a thing* in order to further build composite services [HZHC12].

Finally, a number of European initiatives revolving around the IoT have led to the description of reference architectures [7] and middlewares [8] designed to improve interoperability and creation of services and applications.

6. Linking Open Data Cloud, `http://richard.cyganiak.de/2007/10/lod/`
7. IoT-a, Internet of Things Architecture, `http://www.iot-a.eu`
8. The Linksmart middleware (previously known as the Hydra middleware), `http://sourceforge.net/projects/linksmart/`

Analysis

While a lot of various ontology-based models and semantic middlewares have been designed in the area of modelling context information as well as pervasive environments and their constituents, all of them come with shortcomings if replaced in the scope that this dissertation addresses.

In particular privacy policies tied to users, connected devices and applications are only fully covered in [SPL06] and partially covered in [Che03, GS04].

While associating location information to a user or a connected device is tackled in many approaches [Che03, GS04, WZGP04, SPL06] none of the approaches defines a formal location model enabling e.g., to deduce if a device is in the vicinity of a user. In other words, if some approaches allow to say that 'a TV is located in room 123', none is capable of determining if this 'TV is close to room 456' because none provides a model enabling to localized 'room 123' from 'room 456' [9].

Temporalities implied by the mobility of the constituents of IoT systems are covered in some works through annotations. A consequence is that additional reasoning procedures – w.r.t what OWL and rules offer – must be programmed e.g., to deduce the availability of a connected device or an application at a given time.

Another important weakness associated to these studies is their dependency to OWL or rules reasoning. Similar to the argument pointed by Henricksen that these approaches are unable to deal with uncertainty, returning results to a query implies that all requirements expressed in this query be satisfied. In other words, in a context where one is searching a connected device with some requirements; answers are provided if and only if the whole set of requirements are handled. This 'all or nothing' approach has however drawbacks, especially as it precludes distinguishing between prominent and negligible requirements to provide sufficiently good answers.

Another weakness of these models lies on the lack of support for user preferences and user profiles. While personalized results based on privacy policies may be returned to users in [Che03, GS04, SPL06], none of the approaches is capable of appreciating results in function of other user criteria such as his potential disabilities, his age, etc. Concurrency accesses inherent to offering connected devices to multiple users are also not covered in any of these works.

9. Note that the IoT-A project provides support for such a modelling, as a result of being one of the projects in which this dissertation has contributed, see IoT-A deliverable D4.3 (Section 3.2.2), `http://www.iot-a.eu/public/public-documents/documents-1/1/1/copy2_of_d4.2/at_download/file`

Finally, all the approaches providing query support and reasoning capabilities do not pay attention to the way such reasoning should be performed. In particular, reasoning is performed in a dedicated place and has to deal with all the modelled information. The rise of mobile connected devices as well as the plurality of applications and users calls however for localized and interconnected reasoning procedures to avoid potential scalability problems.

2.2 Searching through semantic similarity measures

As mentioned in the previous chapter, we believe that establishing semantic similarity measures to support the discovery of connected devices in smart environment is a key requirement. As the topic is widely covered by the Semantic Web community, this section provides a review of the most well known measures and places them in the context of concepts described in ontologies almost devoid of instances and underlied by the DL $\mathcal{SHOIQ}$. Choice of this DL comes from the complexity of the models established in the Chapter 3 of this dissertation. Amongst the various possibilities to classify the existing semantic similarity measures, one is to consider two groups: the extensional-based and the intentional-based approaches.

2.2.1 Extensional-based similarity measures

The extensional-based similarity measures use the extensions of the concepts that they compare (i.e. their instances) to determine a similarity value.
In this category, many methods have considered the overlap of the extensions of two concepts being compared in order to produce similarity measurements. Grounded by the work of Jaccard [Jac01], D'amato et al. [dFE05] proposed a semantic similarity measure using the ratio of instances belonging to the intersection of the concepts that they compare with the number of instances belonging to their union. Resnik [Res95], proposed a slightly different approach, defining the semantic distance of two concepts in terms of the IC (Information Content) conveyed by their LCA. To compute the IC of a concept, Resnik used the work of Ross [Ros09] i.e. where the Information Content is quantified as a negative log likelihood ($\text{IC}(C) = -\log(p(C))$ and where $p(C)$ is a probability associated to the concept C. In his work, Resnik defined $p(C)$ as the probability of occurrence of C in a corpus. Further works done by Jiang & Conrath [JC97] as well as Lin [Lin98]

have extended Resnik's vision, by considering the variation between the IC conveyed by the concepts being compared and the IC conveyed by their LCA, to obtain a semantic similarity. In these works the probability associated to any concept C accounts for the fraction of the instances belonging to C divided by the total number of instances in the ontology.

Another set of methods were developed to cope with the semantics of ontologies underlied with expressive DLs (at least DLs up to $\mathcal{ALC}$). In particular, D'amato, Fanizzi et al. [dFE05, dFE06, FDE08] proposed several measures eliciting the semantics of any concepts defined in such ontologies. In each of their works, the concepts of the ontology are rewritten in their normal forms and a semantic similarity measure is proposed, relying on the overlap of the extensions of the concepts being compared.

The main drawback of these methods however, is that they do not correctly interpret the semantics underlying any two concepts that they compare. In particular, they consider that two concepts are totally dissimilar if they do not share any instances, whatever they contain common semantics or not (see [DSF08] for a discussion on this point).

To cope with this limitation, D'amato et al. [DSF08] proposed to compute the semantic similarity of ontological concepts by relying on the variation between the number of instances in the concept extensions and the number of instances in the extension of what they call the "good common subsumer" (GCS) of the compared concepts. The use of the GCS – a variant of LCS but for concepts expressed through operators of the $\mathcal{ALE}$ DL – allows this measure to take into account the semantics underlying the concepts being compared, while the computation of the variation of the extensions (instead of the their overlap) allows to address the aforementioned drawback. However, this measure has two other limitations. First it does not bring much difference in the case of ontologies underpinned by a DL above of $\mathcal{ALE}$ and in particular does not take into account the special features s of DLs allowing hierarchy of (possibly) transitive roles. Second, it assumes that the ontology covers both the concepts and (at least) a large portion of the instances of a given domain, which again is not always the case. To summarize, we believe that this measure should be chosen in the case of ontologies defining both concepts and instances of a given domain and where the semantics underlying this ontology is based on a DL up to $\mathcal{ALE}$. Experimentations performed in Chapter 6 allows to appreciate how this measure fails in the case of ontologies devoid of instances or underlied by an expressive DL such as $\mathcal{SHOIQ}$.

2.2.2 Intentional-based similarity measures

Unlike extensional measures, intentional measures focus on the structure of the concept definitions in order to evaluate their similarity. In this category, a lot of semantic similarity measures have been proposed as functions of the *path distance* between concepts in the hierarchical graph underlying the ontology. In this context, Rada et al. [RMBB89] proposed that the semantic distance of two concepts accounts for the length of their shortest path (given by their LCA). Leacock & Chodorow [LCM98] further transformed this distance to a similarity, considering the highest path length existing in the hierarchical graph. Wu & Palmer [WP94] considered the computation of the path between the $\top$ element of the hierarchical graph and the LCA of both compared concepts. Finally, Ganjisaffar [GANJ06] set weights on each concept of the graph based on the maximum length of the existing paths between this concept and the $\top$ concept and further used the LCA of two compared concepts to get a similarity measure. All these four measures, have in common that they only consider the hierarchical graph of concepts to compute semantic similarity measures. As a consequence, they can be equally applied on ontologies whether they contain instances or not. However, these methods share the point that they are not able to evaluate the semantics that underlies a concept. As relying on the hierarchical graph of concepts, these methods give the same similarity value to any pair of concepts sharing the same LCA, whatever these concepts share some common semantics or not.
Other works proposed to consider all possible relations (i.e., in $N_R \cup \{\sqsubseteq\}$) between any two concepts to compute their similarity. In this context, Sussna [Sus93] used weights on all edges linking two concepts to compute a semantic similarity distance. Although this method computes a different "shortest path" between two concepts, it does not allow any further to consider the semantics that they convey. In 2011, Pirro et al. [PE10] proposed to assign a score of informativeness to each concept by using the whole set of relations defined in the ontology. For each concept definition, they defined an extended Information Content (eIC) consisting of the sum of the intrinsic Information Content (iIC) [SVH04] applied on each property in the definition of the concept. Computing the iIC of a concept involves the ratio between the number of sub-concepts in the hierarchical graph and the number of concepts defined in the ontology. This method, however, has the same drawbacks than the aforementioned ones, that is, it is not able to convey the semantics underlying any concept as it does not consider the type of restrictions (e.g. universal, existential, on cardinality, etc.) applied to the properties in the definition of a concept.

As a consequence – and as stated by the authors – their work does not cover ontologies underlied with expressive DLs.

Another method [MS02] focused on the problem of ontology alignment and proposed a semantic similarity method based on the taxonomic overlap of two hierarchies of concepts. In this work, Maedche et al. used the semantic cotopy of a concept. The similarity of two concepts is then a function of their semantic cotopy, i.e. the ratio between the intersection of the two semantic cotopies and their union. Again as mentioned in [DSF08], this measure strongly depends on the hierarchy of concepts and thus, does not capture the semantics conveyed by concepts defined with an expressive DL. Based on the works of D'amato [dFE06], Janowicz et al. [Jan06] proposed a method based on the syntactic definitions of the concepts. Their method however, does not allow to recognize equivalent concepts and as a consequence does not return a similarity equals to 1 as soon as two equivalent concepts are written differently. Janowicz et al. further refined this work in [JW09] by proposing a method using a modified tableau algorithm (compared to [HS05]) as well as alignment matrices. In this second work, they reduced the similarity of two normalized concepts expressed with a DL $\mathcal{SHI}$ to an inter-instance computation problem. As a consequence, their method requires the presence of instances in order to compute similarities. Moreover, their works do not handle the specific features of a DL containing nominals.

Chapter 3

Defining models to support mobile users

In this chapter we motivate our choice of using Semantic Web technologies to define models for connected devices and applicative requirements. We continue by presenting the different concepts defining the representations of connected devices and applicative requirements. In particular, these concepts cover the following aspects:

— Standardizing the functional behavior and utilized structures of connected devices
— Human-oriented representation of connected device capabilities
— Access rights and time constraints associated to connected devices
— Spatial configuration of an indoor smart environment
— Expressiveness of applicative requirements

We also present our thoughts that led to searching how to design user profiles. In particular, we explain our choice of proposing a PhD thesis based on the investigation of fuzzy logic in order to cope with the limitations pertaining to the sole use of Semantic Web technologies when representing user profiles.

We conclude this chapter by proposing what we think to be the minimal Description Logic underlying the semantics of connected devices, allowing further the design of a semantic similarity measure detailed in the following Chapter 4.

Ideas exposed in this chapter have been presented in a related way in publications written over the last 3 years in which we proposed a semantic representation of Web-enabled devices [Chr11, CBL+11] and applications [CBTB12] possibly dispatched in a modelled indoor environment [Chr12].

3.1 Rationale in using Semantic Web technologies

The previous Chapter 2 highlights the weaknesses of the diverse existing models to support mobile users roaming across different smart environments, calling for the definition of new ones. An open question, however, lies on the selection of an appropriate formalism to provide models. While the weaknesses of CML and ORM based approaches (also pointed in the previous Chapter 2) led us to privilege Semantic Web technologies, our choice was emphasized by the conclusion drawn by Lassila [Las05], considering that *the use of Semantic Web technologies was particularly well suited to improve interoperability as enabling a rich and flexible representation of any resource able to be described.* More specifically, this choice has been made for three reasons:

— The ability to describe a connected device and an applicative requirement through a set of logical axioms capturing most of their respective semantics.
— The necessity for a search engine supporting mobile users to exploit 'processable' descriptions,
— The ability to design semantic similarity measures leading to an efficient selection of connected devices when requested.

Thus, the resulting descriptions benefit from the following:

An identification scheme: In the Semantic Web, all entities defined in a model are referenced by URIs.

Extensibility: This is one of the main concepts of the Semantic Web illustrated through OWL. The Open-World Assumption (OWA) states that everything not explicitly stated in a model is undetermined. This concept allows integration of additional models to refine connected device descriptions. According to the opposed Closed-World Assumption (CWA) something that is not explicitly stated does not exist. In this vision, the whole set of connected devices (resp. applicative requirements) would have to be modeled within itself, constricting extensibility.

Ability of domain-driven models to be interlinked: Although a particular domain is commonly represented by a model, it is possible for every entity composing such a model to reference other entities from a different model by using arbitrary relations. Therefore a decentralized, dynamic and extensible collaborative information space can be built.

Use of standardized languages: The use of standardized languages enables computers to automatically read and interpret information so that applications or programs can

gather the desired information from different sources in a generic way.

Model expressiveness: The use of Description Logic allows Semantic Web based engines to infer logical consequences from a set of asserted facts or axioms. In particular, one of the major task of a semantic engine is to classify the concepts defined in the ontology. Leading to a hierarchical graph composed of super/sub-concepts, procedures can be developed in order to find semantic similarities between concepts, helping to support mobile users to find (partially) equivalent resources when roaming across smart environments. As an illustration, a reasoning procedure would propose either a phone or a loudspeaker as devices able to 'emit a sound' just because both of them have a description covering all what is required to 'emit a sound'.

3.2 Modelling connected devices

From a computer scientist viewpoint, the most obvious way to start the representation of a connected device is to speak about the functionalities that it offers. In particular, functionalities are likely to be associated with a device state, this state changing according to the different actions performed on the device. Grounded by initiatives such that [MBH+04] who considered services as processes and [GD06] who described the semantics of Petri nets, the representation of connected device comprises concepts allowing to describe its associated finite-state machine. In particular, we consider a device as constituted of states, each state giving access to a set of functionalities. Each functionality is primarily considered as realizing a capability, i.e. a service as understood by an end-user (detailed in Section 3.4 as underlying the modelling of applicative requirements). In its simple form, we represent the State and Functionality concepts by the logical descriptions detailed by Equation (3.1) and Equation (3.2).

$$\text{State} \sqsubseteq \geq 1 \text{hasFunctionality.Functionality} \tag{3.1}$$

$$\text{Functionality} \sqsubseteq \geq 1 \text{ realizes.Capability} \tag{3.2}$$

These definitions allow linking functionalities with states and, as such, enable a first level of deductions to be performed: the functionalities available in a state as well as the set of states giving access to a particular functionality.

Building the FSM of a connected device further accounts for the ability to describe the

transitions that happen when triggering functionalities of such device. As such, we define the Transition concept by the following Equation (3.3) conveying the fact that actuation on any functionality may possibly change the state of the device. In this definition, we consider a functionality as always triggered from an initial state and always reaching a final state. Note that both states are not necessarily different. In our approach, defining the semantics of the Transition concept also accounts for ensuring that two transitions described with the same incoming state and the same functionality necessarily go to the same outgoing state [1]. Additionally, these two transitions will be considered as equivalent. To enforce such condition, the Transition concept definition is completed by associating a key (as defined in OWL) on the properties hasFunctionality and hasIncomingState. The semantics of this key, is defined in Equation (3.4).

$$\begin{aligned} \text{Transition} \quad \sqsubseteq \quad & = 1 \text{ hasIncomingState.State} \sqcap \\ & = 1 \text{ hasOutgoingState.State} \sqcap \\ & = 1 \text{ hasFunctionality.Functionality} \end{aligned} \tag{3.3}$$

$$\begin{aligned} & \forall x, y, f_1, s_1 \text{ such that:} \\ & x, y \in \text{Transition}^{\mathcal{I}^2}, \\ & (x, f_1) \in \text{hasFunctionality}^{\mathcal{I}} \wedge (y, f_1) \in \text{hasFunctionality}^{\mathcal{I}} \\ & (x, s_1) \in \text{hasIncomingState}^{\mathcal{I}}, (y, s_1) \in \text{hasIncomingState}^{\mathcal{I}} \\ & \text{then } x = y \end{aligned} \tag{3.4}$$

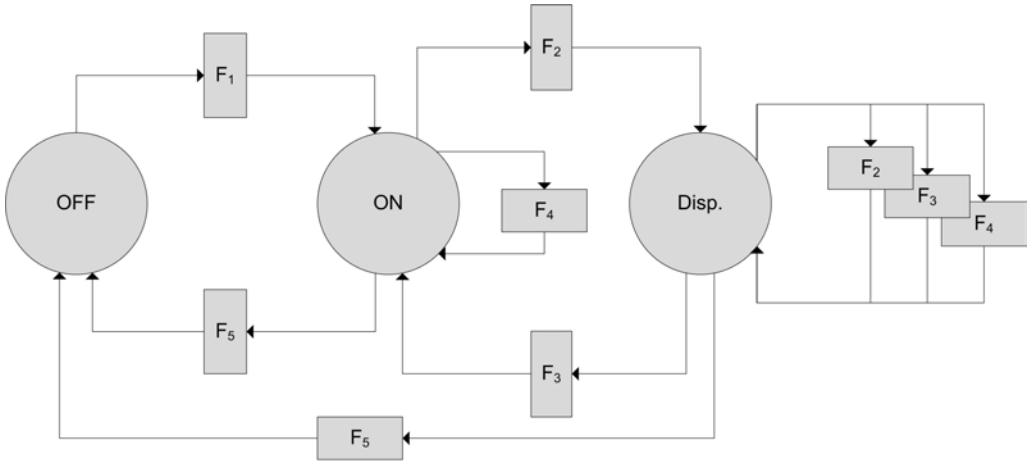
As an illustration, an excerpt of the representation of a connected TV could be represented by the statements displayed in Table 3.1 (three states, three functionalities, four transitions with three of them keeping the device unchanged from a state point of view). Describing all possible transitions would allow building a simple graph accounting for the FSM of the TV (see Figure 3.1).

While we kept the representation of the FSM of a connected device as simple as possible, additional information are required in order to sufficiently describe any connected device. Indeed, the previous three definitions solely participate to build a static representation of the device and do not represent e.g., who can access a functionality, if particular roles or access rights are required or if the connected device can support simultaneous accesses.

1. Note that different approaches may be adopted e.g., one taking into account the values of the parameters consumed by the functionality being triggered in a given incoming state to decide whether it always reaches the same outgoing state or not

$\{\text{ON, OFF, Displaying}\} \in \text{State}^{\mathcal{I}^3}$
$\{\text{TurnOn, TurnOff, PressButton1, PressButton2, IncreaseVolume}\} \in \text{Functionality}^{\mathcal{I}^5}$
$\{\text{T1, T2, T3, T4}\} \in \text{Transition}^{\mathcal{I}^4}$
$\{(\text{T1, OFF}), (\text{T2, ON}), (\text{T3, Displaying}), (\text{T4, Displaying})\} \in \text{hasIncomingState}^{\mathcal{I}^4}$
$\{(\text{T1, ON}), (\text{T2, Displaying}), (\text{T3, Displaying}), (\text{T4, Displaying})\} \in \text{hasOutgoingState}^{\mathcal{I}^4}$
$\{(\text{T1, TurnOn}), (\text{T2, PressButton1}), (\text{T3, PressButton2}), (\text{T4, IncreaseVolume})\} \in \text{hasFunctionality}^{\mathcal{I}^4}$

Table 3.1 – An excerpt of the representation of a connected TV



5 Functionalities	3 States	9 Transitions
F_1 = Turn On	OFF	$T_1 = (F_1, \text{OFF}, \text{ON})$
F_2 = Press "1"	ON	$T_2 = (F_2, \text{ON}, \text{Disp.})$
F_3 = Press "2"	Disp. (Displaying)	$T_3 = (F_3, \text{Disp.}, \text{Disp.})$
F_4 = Turn Up Volume		$T_4 = (F_4, \text{Disp.}, \text{Disp.})$
F_5 = Turn Off		...

Figure 3.1 – Example of the finite state machine associated with the TV description

In this dissertation, we believe that accessing a functionality of a device will likely depend on the context in which the device is used. In particular we foresee four situations affecting the determination of whether a functionality of a device is available or not:

— taking into account access right policies (accessing only the functionalities a user has been granted to)
— distinguishing the current user from other authorized users (A user currently watching a TV is able to switch between different programs vs. other authorized users temporarily not granted to access such functionality)
— aligned with the previous case, determining the kind of simultaneous accesses that can be supported (while aforementioned authorized users may not be able to switch between programs they could still be able to trigger the recording of their favourite drama)
— distinguishing between the roles of different users (prioritizing the owner of the device compared to any other user, managing the permissions of the operator of the device vs. the beneficiary of the device, etc.)

The third case is of utmost importance since that unlike Web services that can be spawned in several instances to allow multiple accesses; each connected device is unique due to its footprint on the real world. In this context, concurrent accesses may become problematic and must be taken into account. Specifically, while simultaneous accesses on some connected devices may be perfectly fine (e.g., readings on the value generated by a sensor may be done in parallel), other cases may lead to offering a degraded service to users (e.g., one user turning on the first channel on a TV while a second user turning the TV off).
Undoubtedly, representing the functional behaviour of a connected device accounts for modelling more complex relations where (at least) states, functionalities, access right policies and user roles are tightly intertwined. Taking the form of n-ary relations, these definitions require to model additional concepts.
First, the role of the user must be defined. In particular in this dissertation we distinguish between five different user roles defined in the hierarchy described by Table 3.2. Each role of the hierarchy is defined as a nominal concept (all nominals asserted as different individuals). The choice of using a hierarchy is the ability to infer, e.g., that the owner of a device is a specialization of a user currently using a device itself specializing an authorized user. Thus, such hierarchy enables to position access rights and to determine if a user as a sufficient role to access or not a functionality. Note that we distinguish between the

Owner	$\sqsubseteq$	Operator	Owner	$\equiv$	$\{\text{owner}\}^{\mathcal{I}}$
Operator	$\sqsubseteq$	Beneficiary	Operator	$\equiv$	$\{\text{operator}\}^{\mathcal{I}}$
Beneficiary	$\sqsubseteq$	CurrentUser	Beneficiary	$\equiv$	$\{\text{beneficiary}\}^{\mathcal{I}}$
CurrentUser	$\sqsubseteq$	AllUsers	CurrentUser	$\equiv$	$\{\text{using}\}^{\mathcal{I}}$
AuthorizedUsers	$\sqsubseteq$	UserRole	AuthorizedUsers	$\equiv$	$\{\text{authorized}\}^{\mathcal{I}}$
$\{\text{owner}\}^{\mathcal{I}}, \{\text{operator}\}^{\mathcal{I}}, \{\text{beneficiary}\}^{\mathcal{I}}, \{\text{using}\}^{\mathcal{I}}, \{\text{authorized}\}^{\mathcal{I}}$ pairwise differents					

Table 3.2 – Hierarchy of roles. Each role is defined as a nominal. A role includes in another accounts for a role with higher accesses.

owner of a device and the beneficiary of the device, to model situations where e.g., an electrical company owns and operates an electricity meter that benefits a user. Defining the roles of users enables to manage simultaneous as well as restricted accesses. Indeed, any transition of a connected device can now be scoped (contextualized) with one role allowing to say that in a particular state, the functionality is restricted to users with some roles. Representing such a context accounts for specializing the aforementioned Transition concept and is detailed in the Equation (3.5)

$$\text{ScopedTransition} \sqsubseteq \text{Transition} \sqcap = 1\ \text{requiresRole.UserRole} \tag{3.5}$$

Following on the example of TV, suppose that a user switches the TV on channel 1, leading to the State “Displaying”. The association of a UserRole on each of the functionalities associated to this State could lead to the following rights: “Turning OFF the TV” granted to the current user (and according to the hierarchy of roles, granted to the owner, the operator and the beneficiary of the device), “Accessing the backlogs of the TV” granted to the operator and e.g., “Turning up and down the volume” granted to all authorized users. Aside to the roles that users can hold, access rights must also be defined to further give information about who can access what. In particular, in one case, one user may be granted to access to some functionalities of a first connected device, while it may be denied to access to functionalities of a second one (both possibly possessed by a same second user). Grounded by studies having already investigated how to set up access rights to users using ontologies and rules [LZWQ05, mAYKGS09], we propose to define user rights by the patterns shown in Equation (3.6) that associate to any user a set of rights

consisting of tuples linking transitions to a required role.

$$\begin{aligned} \text{User} &\sqsubseteq \exists\ \text{hasRight.Right} \sqcap \\ &\quad\ \forall\ \text{hasRight.Right} \\ \text{Right} &\sqsubseteq =1\ \text{hasRole.UserRole} \sqcap \\ &\quad\ \geq 1\ \text{hasTransition.Transition} \end{aligned} \tag{3.6}$$

Note that to further enable the support of authentication protocols such that WebID [2], the concept of User defined in Equation (3.6) may be defined as equivalent to the concept of Agent defined in the Friend Of A Friend ontology [3].

Another aspect characterizing connected devices relies on the temporal relations that can be associated to them. Indeed, a connected device can be "busy all the day for maintenance purpose", can "become unavailable between 5pm and 9am", can "be used only after another one", etc. A possible result of such time constraints is that they can impact the functional behaviour of the device (as possibly changing over the time). To allow modelling these constraints, this dissertation relies on the work of Batsakis et al. [BP10] that proposed an ontology to describe qualitative time relations, based on the thirteen relations described in Allen's theory[All83].

In details, the ontology proposed by Batsakis et al. is grounded by the works of Welty et al. [WF06] who proposed to represent time relations in an ontology using a *four-dimensionalist approach*. The work of Welty et al. relied on the problem of representing fluents – a term defined by Mc Carthy et al. [MH69] as a function mapping objects and situations to truth. Considering a fluent as a relation that holds within a certain time interval [All84] (with defined starting and ending points of time), Welty's work was about modelling diachronic facts, i.e., facts regarded as having developments in time. Instead of using a reification-based method, Welty proposed to consider any entity of an ontology as having temporal parts, each of these parts being given different properties to represent the entity *at an interval of time*. In this 4D-view, any entity having temporal parts can be seen as a *spacetime worm where temporal parts are slices of the worm* [Sid03]. Such a spacetime worm is illustrated by Figure 3.2. Batsakis et al. further extended the work of Welty et al. with qualitative temporal relations holding between time intervals whose starting and ending points are not specified. In particular, they introduced the

2. WebID, `http://www.w3.org/wiki/WebID`
3. Friend Of A Friend defines relationships between peoples, `http://xmlns.com/foaf/spec/`

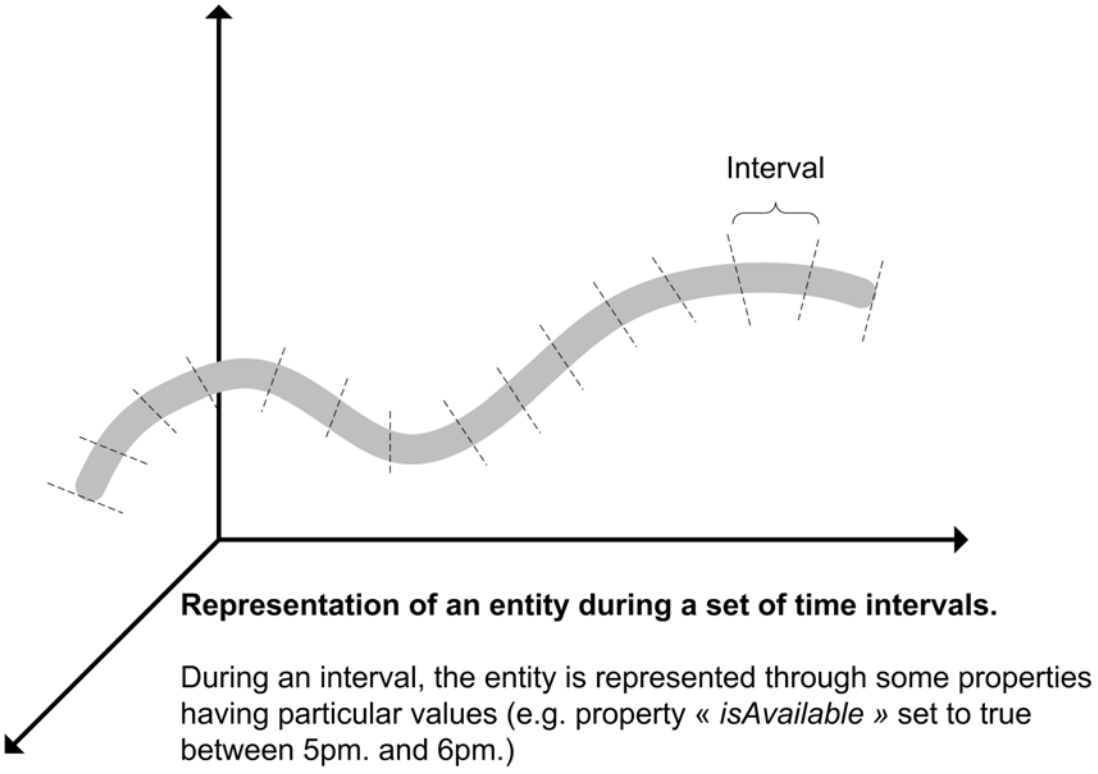


Figure 3.2 – An entity with temporal parts as a spacetime worm

thirteen spatial relations described in Allen's temporal theory from which additional time inference can be performed [All83] using rules. Figure 3.3 illustrates the thirteen Allen's time relations, while Equation (3.7) shows the main definitions of the concepts (coming from works of Welty) that we have included in the description model of the connected devices. Note that unlike Batsakis, we provide seven additional rules allowing to position any pair of time intervals in the classification of Allen's time relations. All these rules are presented in Table 3.3 that also describes six Allen's temporal relations as inverses of six others. Finally, Table 3.4 shows rules expressing what is learnt when composing different Allen's relations. Note that some are also defined in Batsakis work and that in the studies underlying this dissertation we have considered the sole compositions yielding to a unique result (accounting to modelling 71 rules).

$$\begin{aligned} \text{TimeInterval} &\sqsubseteq = 1\ \text{startsAt.DateTime} \sqcap = 1\ \text{finishesAt.DateTime} \\ \text{TemporalPart} &\sqsubseteq = 1\ \text{hasTemporalExtent.TimeInterval} \sqcap \\ &\quad \forall\ \text{hasTemporalExtent.TimeInterval} \sqcap \\ &\quad \forall\ \text{isTemporalPartOf.}\top \end{aligned} \tag{3.7}$$

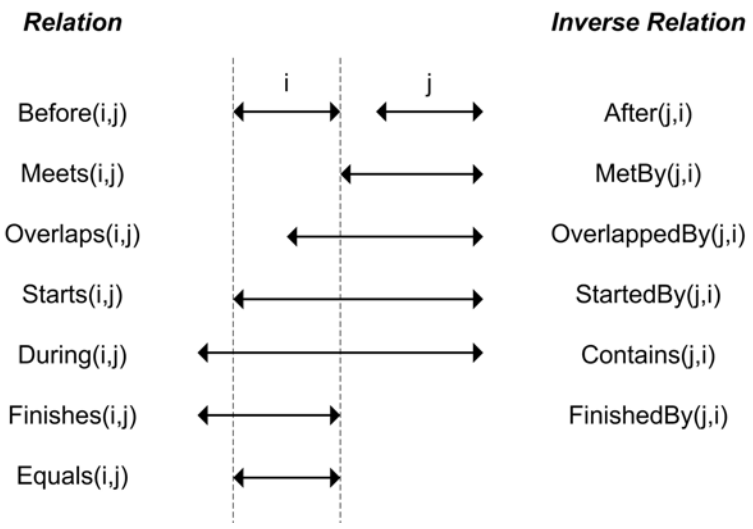


Figure 3.3 – The 13 temporal relations of Allen's theory

Associating temporal constraints to any connected device simply accounts for considering the connected device as the collection of its temporal parts, each of these parts associated to a particular time interval. Thus, while such approach entails declaring a potentially huge set of instances to speak about the same entity, it does not require to create any additional links at the concept level (i.e., there is no need to change the definition of our concepts). As an illustration, the assertions excerpt presented in Table 3.5 shows the transition T_1 (as defined in the previous example of the connected TV) represented through three "slices". The first slice illustrates that T_1 can be granted to any authorized user between 9am. to 5pm. The second slice shows that T_1 is restricted to the current user between 5pm. and 1am. Finally, the last slice shows that T_1 can only be performed by the owner during 1am and 9am. Obviously, unavailability of the transition T_1 accounts for the impossibility for any remote user to use the functionality TurnOn when the TV is off. Allen's combination of relations may also be illustrated in scenarios where functionalities must be composed in a given order to realize some applications, i.e., one functionality being triggered after another one.

The last concepts defined in this section rely on the inputs and outputs associated to any functionality. Accounting for the description of types of content, peripherals, etc. that any functionality may require, generate or use; we propose that these concepts be described through another one called 'Structure', defined by a transitive property called 'isComposedOf' to allow a 'Structure' to be composed of other 'Structures'. Equation (3.8) and (3.9) show the definition of the Input (Output being the same) and the Structure

Before(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), startsAt(y, ys), $greaterThan(ys, xf)$
Meets(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), startsAt(y, ys), $equal(xf, ys)$
Overlaps(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), startsAt(x, xs), startsAt(y, ys), $lessThan(xs, ys)$, $lessThan(ys, xf)$
Starts(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), finishesAt(y, yf), startsAt(x, xs), startsAt(y, ys), $equal(xs, ys)$, $greaterThan(yf, xf)$
During(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), finishesAt(y, yf), startsAt(x, xs), startsAt(y, ys), $greaterThan(yf, xf)$, $lessThan(ys, xs)$
Finishes(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), finishesAt(y, yf), startsAt(x, xs), startsAt(y, ys), $equal(xf, yf)$, $greaterThan(xs, ys)$
Equals(x, y)	$:-$	TimeInterval(x), TimeInterval(y), finishesAt(x, xf), finishesAt(y, yf), startsAt(x, xs), startsAt(y, ys), $equal(xf, yf)$, $equal(xs, ys)$
After	$\equiv$	Before^{-1}
MetBy	$\equiv$	Meets^{-1}
OverlappedBy	$\equiv$	Overlaps^{-1}
StartedBy	$\equiv$	Starts^{-1}
Contains	$\equiv$	During^{-1}
FinishedBy	$\equiv$	Finishes^{-1}

Table 3.3 – Allen's concepts defined through rules or as inverse properties

Before(x, z)	$:-$	During(x, y), Before(y, z)
Before(x, z)	$:-$	Overlaps(x, y), Meets(y, z)
During(x, z)	$:-$	Starts(x, y), Finishes(y, z)

Table 3.4 – Learning from composing Allen's concepts in rules

(T1, OFF) $\in$ hasIncomingState$^{\mathcal{I}}$
(T1, ON) $\in$ hasOutgoingState$^{\mathcal{I}}$
(T1, TurnOn) $\in$ hasFunctionality$^{\mathcal{I}}$
{T1@day, T1@evening, T1@night} $\in$ TemporalPart$^{\mathcal{I}^3}$
{(T1@day, T1), (T1@evening, T1), (T1@night, T1)} $\in$ temporalPartOf$^{\mathcal{I}^3}$
$\{(\text{day}, 9am), (\text{evening}, 5pm), (\text{night}, 1am)\} \in$ startsAt$^{\mathcal{I}^3}$
$\{(\text{day}, 5pm), (\text{evening}, 1am), (\text{night}, 9am)\} \in$ finishesAt$^{\mathcal{I}^3}$
{(T1@day, day), (T1@evening, evening), (T1@night, night)} $\in$ temporalExtent$^{\mathcal{I}^3}$
{(T1@day, authorized), (T1@evening, current), (T1@night, owner)} $\in$ temporalExtent$^{\mathcal{I}^3}$

Table 3.5 – An excerpt including temporal representations of a described connected TV

concepts.

$$\text{Input} \sqsubseteq = 1 \text{ isMadeOf.Structure} \tag{3.8}$$

$$\text{Structure} \sqsubseteq \forall \text{ isComposedOf.Structure} \tag{3.9}$$

The transitivity of this property enables then a semantic engine to retrieve all sub-structures of one structure by issuing a unique query.

Note that unlike the other concepts and properties presented in this section, 'Structure' and 'isComposedOf' are abstract concepts that must be specialized e.g., by device providers when describing their devices. For instance, a phone provider may define a 'Call-Input' concept, specializing 'Structure' and having the property 'hasCalleePhoneNumber' specializing 'isComposedOf'. Another specialization related to peripherals could be that of a 'TV' (specializing 'Structure'), using the property 'hasUSBPorts' (specializing 'is-ComposedOf') and linked to a third and four 'USB2.0' and 'USB3.0' structures. This way, resource providers are free to use their own vocabularies (and ontologies) to declare the structures used by their connected devices. The counterpart of this approach is that each connected device may come with a different representation of the structures it requires/generates or uses. To cope with this challenge an additional process in charge of establishing similarities between structures must be defined, using as much as possible the specificities of the DL underlying such representation (see Chapter 4).

Definitions of the concepts detailed in this subsection do not preclude a set of assertional axioms to embed additional facts and should be considered as the minimal requirements to be typed as 'State', 'Functionality', 'Transition', etc. Thus, by providing these details, this model enables users to know which connected device' functionalities are available at a given time, based on their access rights, their roles, the state the device currently is and the context in which the device is used. In the case of programs running applications, this model also allows determining the chain of states that must be crossed in order to access to a given functionality.

The recapitulative Figure 3.4 shows the interlinks between the main concepts defined in this section. In particular, these links arise from the semantics of the concepts. In other words, these links are what are required to conform with the model thus, nothing precludes to have more links, in particular when linking a functionality with some inputs and outputs.

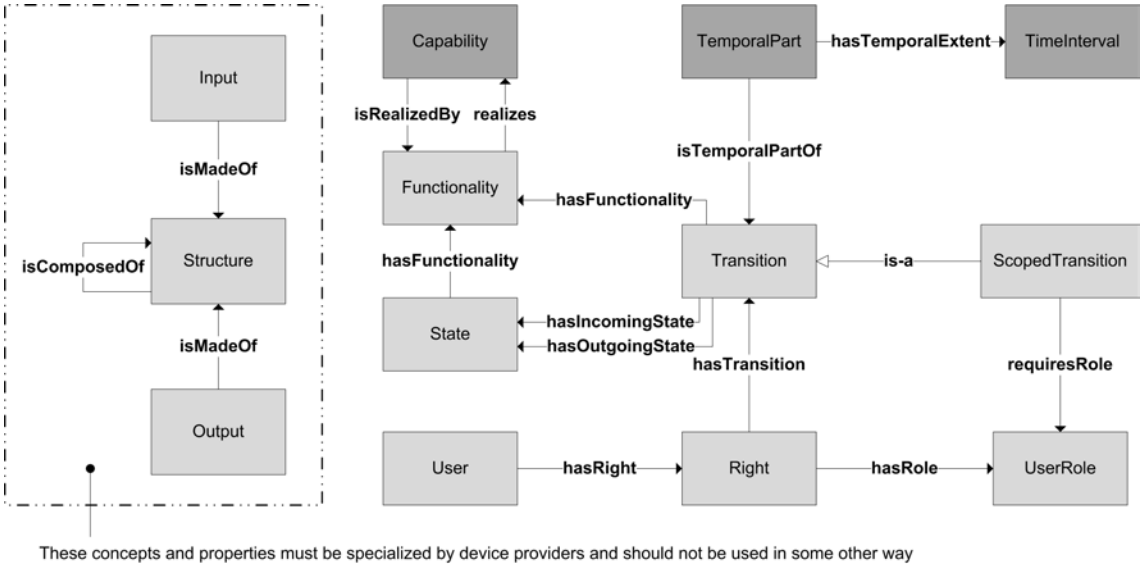


Figure 3.4 – Interlinks entailed by the semantics of concepts

3.3 Modelling the location associated to smart environments

One of the peculiarities of IoT-enabled smart environments is that they involve connected devices, users (and their applications) located outdoors (e.g. temperature sensors deployed within the city) as well as in indoor environments (e.g. different rooms of a building providing a set of connected devices such as light sensors or surveillance cameras). While describing location of outdoor devices can be achieved through WGS-84 [4] or Geonames [5] vocabularies; describing the location of devices, applications or users in indoor environments requires a refined description of the location concept. As an example, consider the building of one company composed of several labs, conference rooms and other premises. One may be willing to create a smart environment in each of these rooms as managing different various kind of connected devices and different profiles of persons. Detecting the neighborhood, i.e. the set of nearest smart environments of a given user in order to propose scalable search mechanisms when configuring his application would then become impossible if relying on the aforementioned vocabularies only. Based on the same thoughts than projects such that CoolTown [KBM+02], we propose to associate location information to any connected device, application and nomadic user. We however take a different approach than Cooltown, by proposing that such location information be underlied by a unified and logical representation model. Revolving around the results exposed

4. Basic RDF Geo Vocabulary, `http://www.w3.org/2003/01/geo/wgs84_pos#`
5. GeoNames, `http://www.geonames.org/ontology/documentation.html`

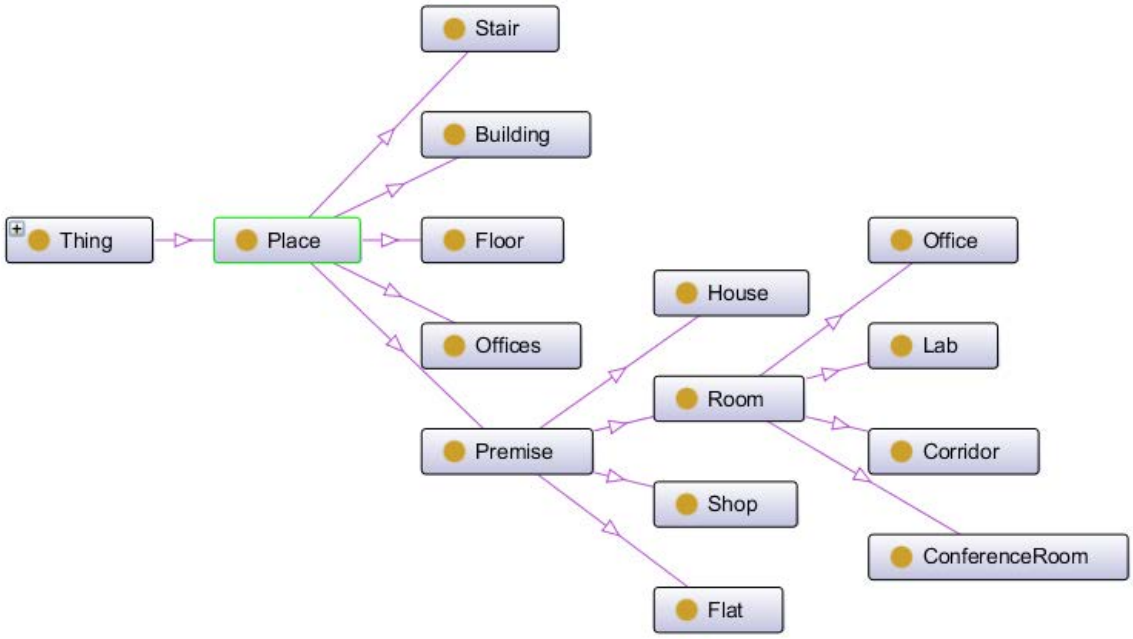


Figure 3.5 – Indoor location concepts

by Randell et al. [RCC92], the indoor location model is composed of various concepts (room, corridor, stairs, etc.) as well as a set of properties denoting possible relationships between instances of the aforementioned concepts (a room *is adjacent to* another, a room *is at the same floor than* another, a room *shares a door with* another, etc.). Instantiating this model accounts for creating a logical representation of the blueprint of a building, enabling e.g., a semantic engine to make some deductions. The main indoor location concepts, represented in Figure 3.5 are then attached to the descriptions of any connected device, any application or to any representation of a nomadic user.

Goals of this model are twofold. The first intent is to provide an efficient search mechanism by taking into account the spatial distribution of connected devices when a nomadic and localized user is searching for a connected device (recall the scenario in the Introduction section). In particular, this information is used to elaborate search strategies in buildings where multiple smart environments are deployed, as detailed in Chapter 5. The second intent of this model is to capture constraints (e.g., stairs) of a building to ensure that users will be able to reach the devices with which they want to interact. In particular, this information is to be coupled with user profiles (defined in the next section) when proposing search results. As an illustration, a search mechanism could overlook some connected devices matching a user functional need because they would entail for such a user to go up

or down some stairs while it was indicated in his profile that the user was in a wheelchair. The resulting model proposes to describe different places gathered under the *Place* concept and representing structures of buildings, rooms, or other premises. Due to the various types of places that may be described, the *Place* concept has a broader meaning that can be narrowed to a building, a floor, a stair or other kind of structures. Some of these concepts are formally defined (based on logical predicates), allowing reasoning tasks to be performed. As an instance, the *Building* concept is modelled as an entity that contains at least some *Floor*. Its formal definition is given by Equation (3.10). A stair is also given a formal representation as connecting at least two floors (see Equation (3.11)).

$$\text{Building} \sqsubseteq \exists \text{ contains.Floor} \tag{3.10}$$

$$\text{Stair} \sqsubseteq \exists \text{ connectsFloor.Floor} \tag{3.11}$$

Along with these concepts, some properties have been defined allowing different places to be interlinked and localized relatively to others (e.g. a *Room* can be *adjacent to* another *Room*). This set of properties, summarised in Table 3.6, provides a small but necessary core of relations between different places enabling e.g., to define knowledge sharing rules such that the ones that will be presented in Chapter 5). In particular, spatial localization and accessibility between different premises are performed by defining rules, shown in Table 3.7. Finally, the properties defined towards others, as well as the properties having specific characteristics (e.g., *transitivity*), are listed in Table 3.8. Note that although this model contains a small set of premises and properties, nothing precludes to extend it with additional concepts. In particular, if the model is instantiated in a language that makes the open world assumption (OWA) such as OWL, it is always possible to extend the knowledge of the model by adding definitions of other types of premises [6]. Besides, more complex relationships between places may be envisioned. Finally, note that the current proposed model assumes that places have a simple geometrical form (we only consider rectangular or circular places) to describe their relative localizations. Additional properties and concepts may therefore be defined in order to take into account places with more complex geometrical form (e.g. torus, L-shaped structures, etc.).

6. OWL can import new ontologies extending its Knowledge Base with additional definitions. This mechanism is done using the *import* tag in the headers of a given ontology, `http://www.w3.org/TR/owl-ref/#Header`

Property Name (Domain, Range) ***Description***
contains (Place, Place) *Allows a Place to contain other places (e.g. a floor containing some rooms)*
isAdjacentTo (Place, Place) *When a place is next to another, e.g. when both share a wall*
inEast (Place, Place) inWest (Place, Place) inNorth (Place, Place) inSouth (Place, Place) *Refinement of isAdjacentTo*
sharesDoor (Premise, Premise) *Means that some premises may have a door in common*
connectsFloor (Stair, Floor) *Means that a stair serves a floor*
requiresRole (Place, UserRole) *Means that accessing a place may require a role* *In particular UserRole is the concept defined in Section 3.2*
hasFloorNumber (Floor, *Integer*) *Further allows to say whether a room is above or under another one*

Table 3.6 – Semantics of properties interlinking places

isIntraFloorConnected(x, y)	:-	shareDoor(x, z), shareDoor(z, y), $DifferentFrom(x, y)$
Two premises opening on the same corridor are connected.		
isInterFloorConnected(x, y)	:-	contains(fx, x), contains(fy, y), connectsFloor(s, fx), connectsFloor(s, fy), $DifferentFrom(fx, fy)$
Two premises contained in different floors both served by the same stair are connected.		
sharesFloor(x, y)	:-	contains(z, x), contains(z, y)
Two premises contained by the same floor, share this floor.		
isLowerThan(x, y)	:-	hasFloorNumber$(x, n1)$, hasFloorNumber$(y, n2)$, $lessThan(n1, n2)$
The floor with the min number is under the other		
isLowerThan(x, y)	:-	contains(fx, x), contains(fy, y), isLowerThan(fx, fy)
The room located in the floor with the min number is under the other		

Table 3.7 – Spatial localization of premises, performed with rules

isUpperThan	$\equiv$	isLowerThan^{-1}
Both properties are also transitive, asymmetric and irreflexive		
contains	$\equiv$	isIncludedIn^{-1}
Both properties are also transitive, asymmetric and irreflexive		
isConnected	$\equiv$	sharesDoor
Two premises sharing a door are connected. *These properties are also symmetric and irreflexive*		

Table 3.8 – Characteristics of the semantics of some properties

3.4 Semantic models for application templates

Although various applications involving different actors have been presented in the illustrative scenario (presented in the Introduction of this dissertation), the pre-requisite they all have in common relies on the ability for expressing and processing requirements and capabilities. In details, users and applications must express what they need through requirements while smart environments with their resources need to express the services that they can provide through capabilities. A reasoning procedure must finally be performed so as to match requirements with capabilities e.g., in order to provide the mobile user with some guidance when configuring an application. In this context, allowing applications to be instantiated with connected devices accounts for defining a formal and processable semantics further shared by the involved stakeholders.
In this section, we propose to model the semantics of capabilities (Section 3.4.1) allowing to use them with the aforementioned Functionality concept. We then propose a bottom-up approach where we learn from different scenarios in order to extrapolate a semantic for requirements (Section 3.4.2). The semantics consists of a set of logic formulas relying on the the model describing capabilities. As these requirements are implied by the user' mobility and are likely to be considered as requests (to find a device, to reconfigure an application, etc.), we propose to encode them in a query language allowing as much as possible an OWL entailment regime. In this perspective, this section further proposes a mapping function to translate our semantic to SPARQL-DL [SP07] queries (Section 3.4.3).

3.4.1 Modelling connected device capabilities for human understanding

In our vision, we make a distinction between the capabilities realized by connected devices and the functionalities that they offer (concept defined in Section 3.2). In particular, we consider that a capability accounts for a mean to represent a goal that a user can understand. Conversely – and as said in a previous section – a functionality deals with the formal representation associated to inputs/outputs/access rights/temporalities, etc. to realize a goal.
To go towards a formal definition associated to a capability, we design a model by conceptually defining a Capability by the following sentence: ***"a Capability has a mandatory action verb, may have multiple modalities to be sensed and may actuate on an object"***. The resulting model is displayed in Figure 3.6 and shows interlinks between

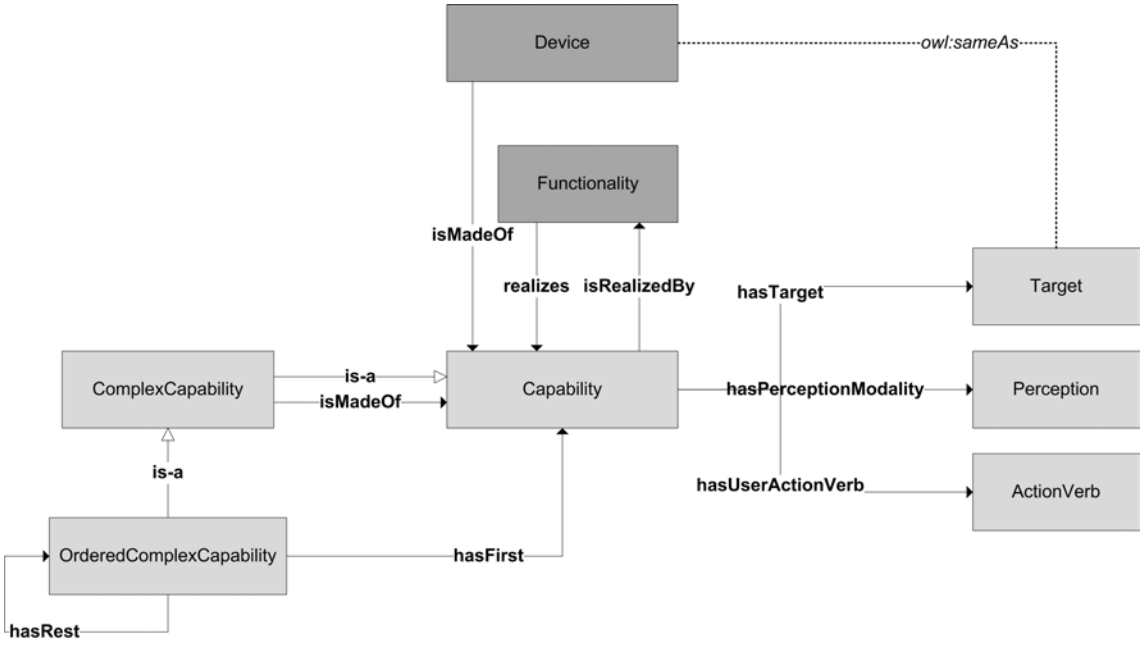


Figure 3.6 – Capability and its relationships with Functionality

connected devices and capability models. In particular, each Functionality is associated with one or more Capability.

Using logical constructs, the formal definition of a capability is given by Equation (3.12), indicating that a capability is equivalent to a mandatory action verb (e.g. “display”, “setup”, “turn on”, “turnoff”) an optional target (e.g. “lamp”, “TV”, “webcam”) and some optional perception modalities (e.g. “can be heard”, “can be seen”). Thus, we consider the case of complex capabilities (i.e. not directly describable with the aforementioned three attributes) and we adjunct the possibility for a capability to be described through a set of (possibly ordered) other capabilities. We define then two additional concepts, ComplexCapability and OrderedComplexCapability to convey this idea (see Equation (3.13)). The difference between these concepts results in considering that in the latter, capabilities need to be realized in a certain oder. In other words, in some case a capability is seen as allowing capabilities to be defined by others (e.g. “blinking” capability defined as the repetition of “provide light – provide dark” capabilities). Thus, such concept enables a semantic engine processing Capability descriptions to supersede one capability by a set of others.

$$\begin{aligned} \text{Capability} \equiv\ & (\ \forall\ \text{hasUserActionVerb.UserActionVerb} \sqcap \\ & \geq 1\ \text{hasUserActionVerb.UserActionVerb} \sqcap \\ & \forall\ \text{hasPerceptionModality.Perception} \sqcap \\ & \forall\ \text{hasTarget.Target}\ ) \end{aligned} \tag{3.12}$$

$$\begin{aligned} \text{ComplexCapability} \sqsubseteq\ & \text{Capability} \sqcap \\ & \geq 2\ \text{isMadeOf.Capability} \end{aligned}$$

$$\begin{aligned} \text{OrderedComplexCapability} \sqsubseteq\ & \text{ComplexCapability} \sqcap \\ & = 1\text{hasFirst.Capability} \sqcap \\ & \forall\ \text{hasRest.OrderedComplexCapability} \end{aligned} \tag{3.13}$$

The aforementioned Equation (3.12) entails the definition of other concepts being respectively *UserActionVerb, Perception* and *Target*.

The *UserActionVerb* concept is formally defined by a set of verbs expressing an action that a user or an application is willing to do. Although containing a fix list of verbs, use of OWL and its "import" mechanism allows easily extending this list with new terms. Note that for our experimentations (detailed in Section 6.2.4, we use a list composed of around 30 verbs resulting from a survey asking users to tell us a list of verbs that they were using to describe a situation involving actions on objects.

The Perception concept is also defined by a list of terms which gather the common five modalities of sensing something: *Seeing, Hearing, Smelling, Tasting, Touching*. Finally, the Target concept conveys the idea that an object may actuate on another one (e.g. actuating on a Lamp by turning on a button switch). Thus, as explained in Section 3.4.2, this concept is required when defining a requirement, in order to further link it with a connected device. Hence, the Target concept is defined as equivalent to connected device (see the use of *owl:sameAs* link in Figure 3.6).

The semantics associated to capability is further used when defining the formal semantics to represent applicative requirements.

3.4.2 A semantics to model applicative requirements

Unlike capabilities, expressing requirements is a hard task due to the various numbers of constraints that they can embed. Some may be expressed with restrictions on the

content they want to get (e.g. recall the Metro Warning application requiring a public transportation feed). Some others may have restrictions on both the functionality provided by a device and its localization (e.g. recall the Local Slideshow application requesting a locally available device able to display pictures) and finally, some others may have restrictions based on some human perception (e.g. recall the Metro Warning requesting a device that can "warn" a user with a signal, whatever the sensory form it might take). Besides, although not highlighted in our scenario, it may be plausible having application templates containing requirements expressed with restrictions on time.

In our view, we consider a requirement as a collection of restrictions that apply on some attributes (functionalities, location, ownership, etc.) associated to a connected device. In other words, requirements can be considered as conditions that a system would check against a set of connected devices. Connected devices respecting all necessary conditions would then be returned for an application template to be configured.

To define the semantics of requirements, we adopt a bottom-up approach divided in three steps. First we gather the different restrictions of the scenario presented in the Introduction of this dissertation. Then, we write these restrictions as logic formulas (using predicate logic constructs) in order to see the types of expressions being formed. Finally, the last step of our method is to deduce the overall (i.e. not constricted to our scenario) semantics tied to requirements by writing a grammar based on the formulas obtained in the second step.

The first step leads to six restrictions: requiring a device consuming (1) or generating (2) a given type of content, located at a particular place (3), realizing a given capability (4), currently available (5) and aligned with access rights (6). Note that a different approach is used in order to ensure that returned results are compliant with restrictions put in the profile of the requester. The second step results in a set of formulas displayed in Table 3.9 and composed of the following:

— Named rules with parameters, e.g. RealizeCapability(x, y, z, t)
— Unary predicates $C(x)$, denoting that x is an instance of the concept C.
— Binary predicates $P(x, y)$, expressing that x and y are in relation with P, e.g. x realizes y
— Material conditional formulas $x \rightarrow F(y, x)$ meaning that if we have x then we must verify $F(y, x)$

ConsumeContent(x, "c")	$\equiv$	CD(x) $\sqcap$ isMadeOf(x, y) $\sqcap$ isRealizedBy(y, z) $\sqcap$ consume(z, "c")
Connected devices consuming a type of content "x" is a variable, CD accounts for "connected device", "c" is an instance of the concept Content and "y, z" are bound variables		
GenerateContent(x, "c")	$\equiv$	CD(x) $\sqcap$ isMadeOf(x, y) $\sqcap$ isRealizedBy(y, z) $\sqcap$ generate(z, "c")
Connected devices generating a type of content		
LocallyAccessible(x, "loc")	$\equiv$	CD(x) $\sqcap$ isLocatedIn(x, "loc")
Connected Devices accessible at a given location "loc". "x" is a variable while "loc" refers to an instance of Location concept		
RealizeCapability(x, "a", "p", "t")	$\equiv$	CD(x) $\sqcap$ isMadeOf(x, y)
	$\sqcap$	"a" $\rightarrow$ hasUserActionVerb(y, "a")
	$\sqcap$	"p" $\rightarrow$ hasPerceptionModality(y, "p")
	$\sqcap$	"t" $\rightarrow$ hasTarget(y, "t")
Connected devices realizing a given capability. "x" is a variable while "a", "p", "t" are values. "y" is a bound variable		
Available(x, "t")	$\equiv$	CD(x) $\sqcap$ isMadeOf(x, y)
	$\sqcap$	isRealizedBy(y, z) $\sqcap$ hasFunctionality(t, z)
	$\sqcap$	isTemporalPartOf(tp, t) $\sqcap$ hasTemporalExtent(tp, ti)
	$\sqcap$	"t" $\rightarrow$ contains("t", ti)
Connected devices available at the time interval "t". "x" is a variable while "t" is a value. "y, z, t, tp, ti" are bound variables		
Available(x, "u", "r", "t")	$\equiv$	CD(x) $\sqcap$ isMadeOf(x, y)
	$\sqcap$	isRealizedBy(y, z) $\sqcap$ hasFunctionality(t, z)
	$\sqcap$	isTemporalPartOf(tp, t) $\sqcap$ hasTemporalExtent(tp, ti)
	$\sqcap$	"t" $\rightarrow$ contains("t", ti)
	$\sqcap$	requiresRole(t, ur)
	$\sqcap$	"r" $\rightarrow$ hasRole("r", ur) $\sqcap$ hasRight("u", "r")
Connected devices accessible by the user "u" with the right "r" at the time interval "t". "x" is a variable while "u", "r", "t" are values. "y, z, t, tp, ti, ur" are bound variables		

Table 3.9 – Expressions of restrictions, using predicate logic

In the aforementioned terms, x and y are either variables, OWL individuals (e.g. a user action verb) or data values. Besides, parameters only appearing at the right-hand side of a rule (i.e. not in the list of parameters of the rule) account for bound variables. Finally, all terms appearing in the left-hand side of a rule are universally quantified. In other words, a term such that isMadeOf(x, y) accounts for "all pairs (x, y) in relation by the property isMadeOf". Although required to build the formulas, the values that these variables can take are not of interest. As an example, the first expression of Table 3.9 is composed of the following terms:

— $CD(x)$ where CD is an OWL concept (representing a Connected Device) while x is a variable. The meaning conveyed is: all x of type CD

— $hasFunctionality(x, y)$, an OWL property with 2 variables x and y. The meaning conveyed is: "all x having the functionality y"

— $consume(y, z)$, an OWL property with 2 variables y and z. The meaning conveyed is: "all y that consume the content z".

Equation (3.14) shows the semantics of requirements that we deduce from above Table 3.9 formulas (with material conditional formulas expressed through disjunctions).

$$\begin{aligned} R(x_1,\ldots,x_m,\ldots,x_n) &= \bigvee_{i=1}^{p} F_i(z_k,z_l) \\ F(z_k,z_l) &= \bigwedge_{j=1}^{q} AF(z_k,z_l) \\ AF(z_k,z_l) &= C(z_k) \vee P(z_k,z_l) \vee MCF(z_k,z_l) \\ MCF(z_k,z_l) &= (z_k \rightarrow P(z_l,z_k)) \\ &= (\neg z_k \vee P(z_l,z_k)) \end{aligned} \tag{3.14}$$

Where:

— $x_1, \cdots, x_m$ are free variables

— $x_{m+1}, \cdots, x_n$ are OWL individuals or values

— z_k, z_l (appearing in $F_i(z_k, z_l)$) are mapped on free or bound variables (x_k)

— $R(x_1, \cdots, x_n)$ is a restriction expressed as a set of disjunctive formulas

— $F(z_k, z_l)$ are conjunctions of atomic formulas

— $AF(z_k, z_l)$ is an atomic formula involving either the OWL type of a given z_k or an OWL property linking a given pair of (z_k, z_l) possibly nested in a material conditional formula.

— $C(z_k)$ is a condition on the OWL type of a z_k (i.e., $z_k \in C^{\mathcal{I}}$)

— $P(z_k, z_l)$ is a condition on an OWL property involving z_k and z_l ((i.e., $(z_k, z_l) \in P^{\mathcal{I}}$)

— $MCF(z_k, z_l)$ is a material conditional formula

As an example, the requirement issued by the Metro Application and consisting of requiring an object outputting content of type "PublicTransportation" would be expressed by the following restriction:

$$\begin{aligned} R(x, \text{“PublicTransportationInfo”}) = {} & CD(x) \\ & \wedge \text{isMadeOf}(x, y) \\ & \wedge \text{isRealizedBy}(y, z) \\ & \wedge \text{generate}(z, \text{“PublicTransportationInfo”}) \end{aligned}$$

$$\begin{cases} x & : \text{a free variable on the set of connected devices (CD)} \\ y, z & : \text{bound variables on the sets of respectively Capability and Functionality} \\ \text{“PublicTransportationInfo”} & : \text{an OWL Individual referring to a Concept specializing} \\ & \text{“Structure” (see Section 3.2)} \end{cases}$$

3.4.3 SPARQL-DL requests for requirements

As seen in the previous subsection, requirements can contain individuals or values. As an instance, the third requirement presented in Table 3.9, is about finding connected devices at a given location where location could refer to places expressed using the aforementioned model of Section 3.3 or using an ontology such that Geonames. While expressing such requirements remains possible in OWL (i.e. defining a LocallyAvailable concept with OWL); two reasons preclude following this approach:

— A very large number of instances that may belong to a concept (regarding the third requirement, there could be as many instances as geo-coordinates possibly formed) requiring heavy computation resources to process them.
— The re-computation of the Knowledge Base, each time a new requirement (represented by a new OWL expression) would be required.

Moreover, the use of disjunctions in the semantics defined in Section 3.4.2 does not allow expressing requirements with SWRL.

$$
\begin{array}{lcll}
\phi : \mathcal{R}equirements & \rightarrow & \mathcal{SPARQL-DL} & \\
 & & \textbf{SELECT } ?x_1, \ldots, ?x_m & \\
 & & \textbf{WHERE } \{\phi(F_1(z_{k_1}, z_{l_1}))\} & \\
R(x_1, \ldots, x_m, \ldots, x_n) & \rightarrow & \textbf{OR WHERE } \{\ldots\} & \\
 & & \ldots & \\
 & & \textbf{OR WHERE } \{\phi(F_p(z_{k_p}, z_{l_p}))\} & \\
 & & & \\
F(z_k, z_l) & \rightarrow & \phi(AF_1(z_{k_1}, z_{l_1})) \textbf{ , } \ldots \textbf{ , } \phi(AF_q(z_{k_q}, z_{l_q})) & \\
C(z_k) & \rightarrow & \begin{cases} \textbf{Type}(?z_k, C) & \text{if } z_k \text{ a variable} \\ \textbf{Type}(A, C) & \text{with A a constant, otherwise} \end{cases} & \\
P(z_k, z_l) & \rightarrow & \begin{cases} \textbf{PropertyValue}(?z_k,\ P,\ ?z_l) & \text{if } z_l \text{ variable} \\ \textbf{PropertyValue}(?X_k,\ P,\ A) & \text{with A a constant, otherwise} \end{cases} & \\
MCF(z_k, z_l) & \rightarrow & \phi(P(z_l, z_k)) &
\end{array}
\tag{3.15}
$$

SPARQL-DL queries can however be formed from this semantics, allowing therefore such requirements to be written down and further processed. Equation (3.15) presents a bijec-

tive function that binds the semantics of requirements to SPARQL-DL expressions. [7]. As an example, Listing 3.2 contains the SPARQL-DL request corresponding to the requirement issued by the Metro Warning application used in the illustrative scenario.

```
PREFIX device: <http://webofdevices.appspot.com/models/device.owl#>
PREFIX cap: <http://webofdevices.appspot.com/models/capability.owl#>
SELECT ?x WHERE
{Type(?x,dev:ConnectedDevice),
    PropertyValue(?x, dev:hasState, ?y),
    PropertyValue(?z, dev:isAccessibleFromState, ?y),
    PropertyValue(?z, dev:realizes, ?c),
    PropertyValue(?c, cap:hasUserActionVerb, cap:EmitSignal)}
```

Listing 3.1 – SPARQL-DL query to be wrapped in an HTTP call

3.5 Representing user profiles

Offering search mechanisms which can deliver meaningful results to nomadic users calls for designing a system capable of interpreting user specificities and preferences. In the context of multiple places of the same building (rooms, floors, etc.) equipped with connected devices, a meaningful result would typically account for not only matching functional requirements, but also entailing an easy access for the user considering e.g., distances separating the user and the device vs. the capability for such user to walk. Interpreting user profiles and preferences however, accounts for the ability to perceive the discriminant pieces of information (e.g., age, health situation, size, role) as well as to understand the meaning conveyed by their values (e.g., meaning of age = *30* or role = *project leader*).
In the past three decades, many studies tried to identify the relevant features characterizing a user by developing learning algorithms as well as formal models. Amongst others, feature-based filtering was a technique developed in order to learn preferences and interests of a user from usage data [PB07]. Collaborative filtering [GNOT92, SKKR01] was another example consisting of comparing similar users to reveal additional interests and preferences. In the context of interests seen as topics and organized hierarchically, domain based inferences were applied on user profiles to infer the probability that a topic be of interest for a given user, based on previously known interests [FK02]. In the field of In-

7. Note that this function does not display the PREFIX headers that should be associated to $R(x_1, \cdots, x_n)$, for readability.

formation Retrieval, studies also emphasized the exploitation of immediate search context to compute user interests [STZ05] instead of using a long-term collection of information. Related to formal representations, various ontological models describing the major categories of user preferences were designed, many of them surveyed in [JT09].

To cope with the challenge of representing user profiles, we launched and supervised a research work – under the form of a PhD – with the goal of determining a set of discriminant categories defining a user. In the scope of our research, these categories were further used in order, for each user looking for connected devices, to filter resources corresponding to its profile. As an example these categories were at the basis of determining the maximal distance that can be travelled by a user when searching for a connected device from a given location.

Amongst the different categories having been determined, it appeared that some of them were taking continuous values e.g., *age of a user* while some other were not e.g., *being in a wheelchair*. While ontologies allow to formally categorize discrete data values (e.g., such that representing the statement "a user has the property *being in a wheelchair* equals to *true*"), they are however not able to qualify attributes having continuous values (such as *age*). Indeed, understanding the meaning of a value depends on a global context more than the intrinsic quantity itself (e.g. in some context the feature *age = 50* accounts for classifying someone as old, while in some other cases it would be classified as young).

We therefore proposed that the research work makes use of fuzzy logic theory, a branch of logic able to handle continuous values.

Formalized by Prof. Lotfi Zadeh in mid. 70's [Zad75a, Zad75b, Zad75c], fuzzy logic theory consists of a multi-valued logic allowing to attach to any "fuzzy" variables a truth-value ranging between 0 and 1. Extending binary sets theory (where variables are either true or false), fuzzy logic is considered as a possibility to apply a more human-like way of thinking in the programming of computers [Zad84], allowing e.g., notions such that "rather X" or "very Y" to be formulated mathematically. In fuzzy logic theory, these notions are represented through *linguistic variables*.

More precisely, following the ideas of [MBKV+02] as well as the formal classification established by Jiang et al. [JT09], we proposed that this work led to the representation of user preferences and interests through a set of features, all of them referring to concepts and properties defined in an ontology. We also suggested that all these features be associated to as many as linguistic variables and that the features having their values in a continuous

interval, be supported by membership functions (their expressions being detailed in the corresponding dissertation [Xu15]).

As an example, this chapter presents six variables deriving from the result of this research work.

— Age
— Body Mass Indice
— Physical activity
— Lower limb pain
— Pregnancy
— Wheelchair

Aligned with these variables, Equation (3.16) shows the semantics associated to the profile of the user, while Listing 3.2 presents an example of user profile, relying on preferences expressed as RDF triples, i.e., processable by a semantic engine. Thus, the example highlights that preferences are represented by properties defined in the user profile ontology.

$$\begin{aligned} \text{User} \sqsubseteq\ & = 1\ \text{hasAge}.\mathit{Integer} \sqcap \\ & = 1\ \text{hasBMI}.\mathit{Integer} \sqcap \\ & = 1\ \text{hasPhysicalActivity}.\mathit{Integer} \sqcap \\ & = 1\ \text{hasLowerLimbPain}.\mathit{Integer} \sqcap \\ & = 1\ \text{isPregnant}.\mathit{Boolean} \sqcap \\ & = 1\ \text{inWheelchair}.\mathit{Boolean} \sqcap \end{aligned} \tag{3.16}$$

```
<owl:NamedIndividual rdf:about="&userprofile;Benoit">                                  1
    <rdf:type rdf:resource="&userprofile;UserProfile"/>                                2
    <hasLowerLimbPain rdf:datatype="&xsd;int">0</hasLowerLimbPain>                     3
    <hasPhysicalActivity rdf:datatype="&xsd;int">2</hasPhysicalActivity>               4
    <hasBMI rdf:datatype="&xsd;int">21</hasBMI>                                        5
    <hasAge rdf:datatype="&xsd;int">34</hasAge>                                        6
    <isPregnant rdf:datatype="&xsd;boolean">false</isPregnant>                         7
    <isInWheelchair rdf:datatype="&xsd;boolean">false</isInWheelchair>                 8
</owl:NamedIndividual>                                                                 9
```

Listing 3.2 – Example of user profile using RDF statements

This approach is aligned with the strategy that we propose in the next Chapter 6 where is defined the overall search strategy aiming to support nomadic users roaming in smart environments. Briefly, the intention of using the aforementioned variables is to propose a

mechanism allowing to filter places where a matching between user requirements and connected device semantics will be performed. Filtering the places accounts for determining the max distance that a user may be able to travel, based on the values of the features found in his profile. Determining the max distance further allows to select a set of places in the vicinity of the localized user.
Interpreting the max distance that a given user will be able to travel, depends on how the values of the different features will be categorized and has also been investigated in the aforementioned research work.

3.6 Conclusions

After having identified the different stakeholders involved in the context of this dissertation, this chapter has analyzed their semantics to further propose representation models. A large part of the modelling is underlied by Description Logics and in particular, when all concepts and rules are interconnected, by the Description Logics SROIQ(D) which, compared to SHOIQ detailed in the Introduction adds the fact that:

— Some roles are combined (which happens in the rules that we have defined) and,

— Some properties have a range defined by a data type (Integer, Boolean, etc.)

The recapitulative Figure 3.7 shows the main concepts of our models, interlinked together by following the different requirements expressed in the diverse equations of this chapter. In addition to the semantic modelling of most of the concepts triggered in this dissertation, the profile of the user is underlied by linguistic variables, making use of fuzzy logic. Once described the representations of connected devices, users and applicative requirements can be processed by a searching procedure, such that the one presented in Chapter 4.

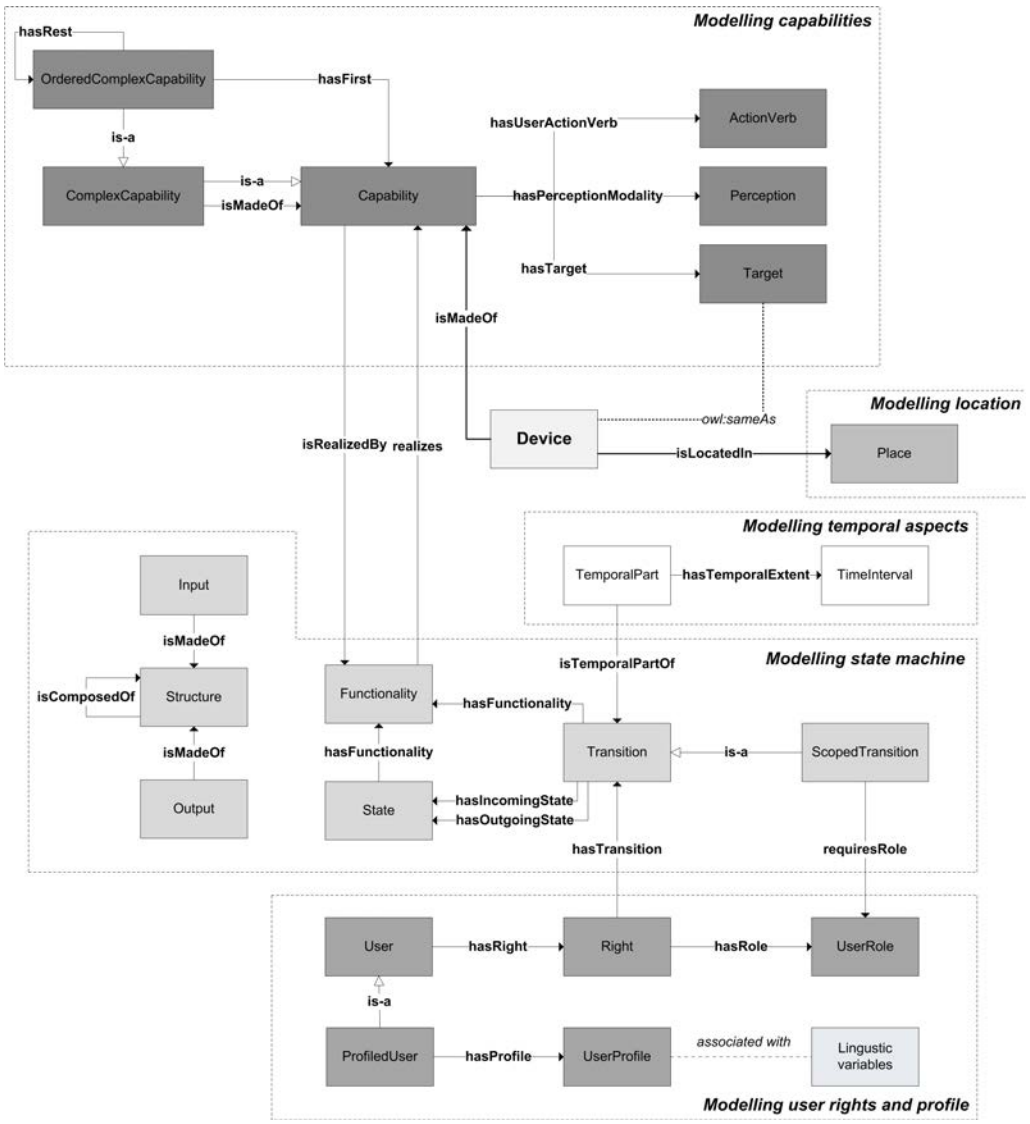


Figure 3.7 – Recapitulative view of the main concepts modelling users, devices and requirements

Chapter 4

Towards producing efficient searching procedures

Together with the definition of a unified representation describing connected devices, users and smart environments; supporting nomadic users roaming across different smart environments requires the development of appropriate searching procedures.
In particular, such mechanisms must deliver customized results, i.e. corresponding to user preferences. They must also take into account user profile and take care of the localization of users and devices, e.g., in order to propose relevant results, as close as possible from (and in any case, accessible to) the user.
Assuming that the user has been reaching an environment (i.e., has been localized) and is requesting some connected devices, we believe that an efficient search procedure results in the two following steps:

— Filtering devices by coupling the profile of the user with spatial and temporal information about devices.
— Amongst the filtered devices, performing an analysis to deduce the ones corresponding best to requirements and preferences expressed in the user' query.

Aligned with the representation models defined in Chapter 3, we believe that such a process can be embodied by performing the steps represented in Figure 4.1.
In this view, the filtering step is performed by using the linguistic variables – recall that they define the user profile – together with the semantic representation of the location concepts in order to define a set of places in which a refined analysis can be performed.
In particular, the linguistic variables can typically be the inputs of a Fuzzy Inference Sys-

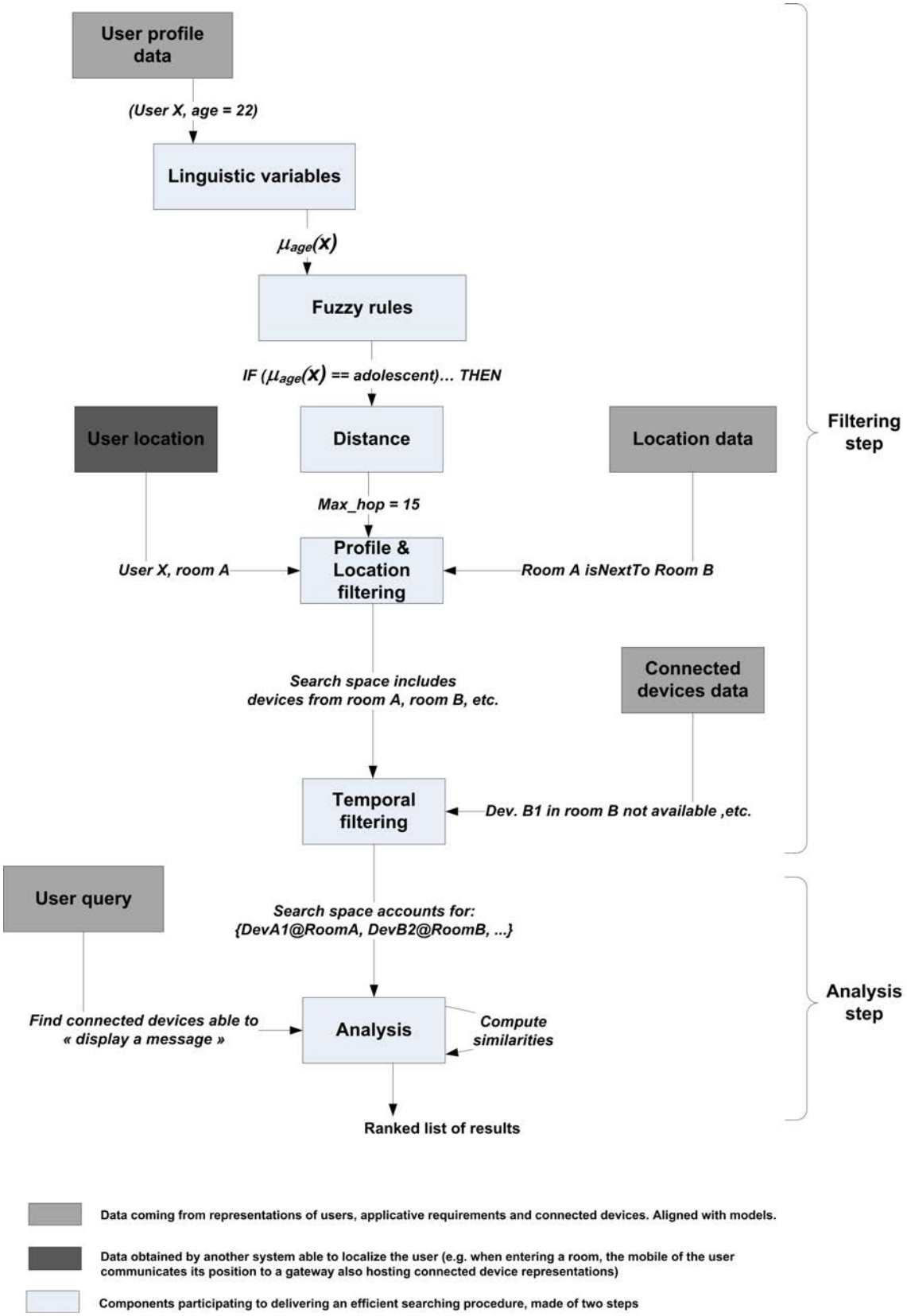


Figure 4.1 – An example of a searching procedure comprising a filtering followed by an analysis

tem (FIS), a rule engine based on IF – THEN clauses and capable of evaluating the values taken by different linguistic variables (i.e. labels having been previously determined using membership functions). Independently of the type of FIS being chosen (Mamdani et al. [Mam77], Sugeno [SY93], Tsukamoto [Tsu79], Jang [Jan93] or Kasabov[KS02]), such a system could be in charge of computing a set of places leading to a set of connected devices (the ones hosted by the determined places). This kind of filtering may also be continued by making use of the temporal constraints associated to the representation of a connected device, mainly to discard those not accessible for a user at a given time.
Representations of these devices are then further used by an algorithm in charge of finding the devices that best answer the requirements expressed in the user' query.
While filtering amongst connected devices does not require particular research and can be done using tools found in the literature, we believe that the open point to offer an efficient search mechanism relies on analyzing how a device answers the requirements expressed in a user' query. In this context, the challenge in proposing relevant connected devices to users accounts for the capability to provide a sound analysis evaluating the similarities between the semantics of a requirement and the remaining semantics of a connected device ($\mathcal{SHOIQ}$).
Accordingly, this chapter focuses on the second step of the searching procedure and details a method performing similarity measurements of any entities described with the DL $\mathcal{SHOIQ}$. The strength of this method is that it is designed to solely rely on the theoretical ground enabling the creation of any ontology underlied by the DL $\mathcal{SHOIQ}$. As a consequence, this method is independent of any "applicative" context and can be used in other areas involving the creation of ontologies up to $\mathcal{SHOIQ}$. This independence is especially emphasized in the experimentations that have been driven, making use of well-known and accepted ontologies available on the Web, all of them addressing diverse domains such as food, wine or biological classifications (details provided in Chapter 6).

4.1 Preamble

Determining the semantic similarity or relatedness of different data (or services) is an important concept in information systems. Usually based on topological or statistical analysis, it is for instance used in information retrieval to expand a query with similar terms [Voo94] or to return ranked results. In information fusion, it is used to reduce uncertainty when merging data from different sources, allowing for instance the creation of an ontology using a bottom-up approach [SM01]. Finally, in data or service maintenance, it allows failure recovery by analyzing which data (resp. service) can be replaced by others. In this chapter, we address the problem of comparing raw data having been associated to concepts [1] or properties defined in ontologies underlied by the expressive DL $\mathcal{SHOIQ}$ (see Section 1.1.1). Usually, these ontologies contain the definitions of a domain of expertise (e.g. automotive, biology, wine, etc.) and may embed few instances of concepts mostly to setup the definition of other concepts (so-called *nominals*) [2]. Generally, designing a semantic similarity assessment process able to compare different ontological concepts consists of defining a measure whose objective is to quantify the *common* (and in the case of [Tve77], *the different distinguishing*) features amongst each pair of compared concepts. Up to now, a plethora of such measures (some surveyed by [Sch08]) has been designed and applied on elements of an ontology, referring to different point of views that one can have about how elements should be compared. Initially, most of the existing methods were based on an adaptation of measures defined for different and less expressive representations (e.g. feature vectors or trees, as mentioned in [d'A]). However such measures are unable to convey the underlying semantics of most ontological representations. Then, some measures coping with expressive ontologies were developed. However, none of them truly address ontologies defined with a DL above than $\mathcal{ALCHN}$. Finally, most of the recent approaches have considered using the extensions of concepts to compute semantic similarity measurements. This assumption however, is problematic when considered ontologies solely define the concepts of a domain (and consequently do not intend to embed instances). Moreover, these methods tend to place any defined concept under the Closed World Assumption (as in these approaches, the concept is assimilated as equal to its set of instances) which, by

1. Using an annotation mechanism such as Semantic Annotations for Web Resources (SA-REST) [LGSpt]

2. In some cases an ontology represents an entire domain of interest i.e., its concepts and instances (e.g. in an automotive context). In some other cases it may focus on representing the concepts and include very few (even none) instances (e.g. the OWL-S ontology)

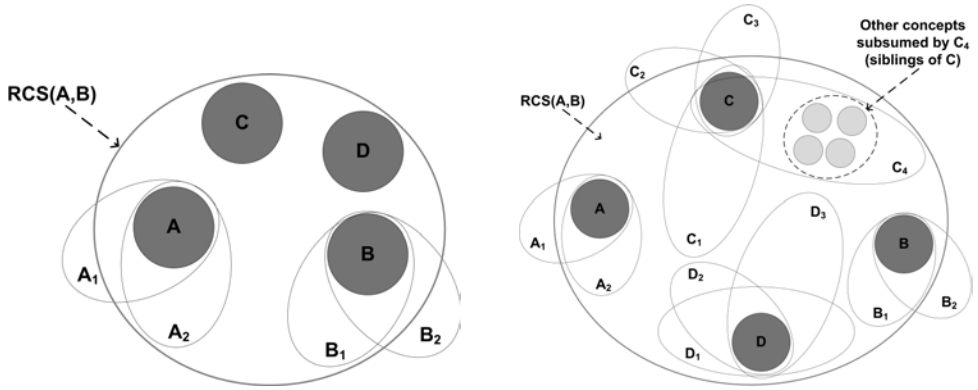
definition, is a wrong assumption when representing concepts in OWL.
To the best of our knowledge, no work has been done in the specific case of ontologies underlied by the expressive DL $\mathcal{SHOIQ}$ where roles can be transitive, inverse of other roles and organized in a hierarchy and where concepts can be defined with instances (nominals). In particular, no work has been done in the context of $\mathcal{SHOIQ}$ ontologies having almost all their concepts devoid of instances. What we do propose then, is a novel semantic similarity measure for such ontologies. Our method relies on an algorithm that expands the classification graph of any ontology underlied by a DL up to $\mathcal{SHOIQ}$ and that weights all the concepts of this expanded graph. Based on these weights, we define a measure that solely relies on the semantics of the concept definitions (i.e. that can work on ontologies devoid of instances) and that is able to respect all the requirements formulated by [DSF08] who theorized a set of criteria that a semantic similarity measure should follow.

4.2 A semantic similarity measure for $\mathcal{SHOIQ}$ concepts

This section introduces a new semantic similarity measure that relies on an algorithm expanding the classification graph of any ontology underlied by the DL $\mathcal{SHOIQ}$ and weighting the nodes of such graph. Based on a $\mathcal{SHOIQ}$ Normal Form (see Section 1.2) and a family of generative functions, the algorithm uses various subsumption properties (see Appendix A for details) taking into account the specificities of the DL $\mathcal{SHOIQ}$ to generate a set of pseudo-concepts (denoted by $\mathcal{PS}$ in the rest of the manuscript, see also Section 1.2) allowing a semantic Web engine to determine additional inferences when reclassifying the ontology. Although useless from the domain definition point of view, these pseudo-concepts are useful when computing semantic similarity as making explicit the underlied semantics implied by the concepts originally defined. All the concepts of this expanded graph are further weighted using their in-degree as well as the weight of their children (0 if the node is a leaf). These weights are further used when computing the semantic similarity of any two concepts, in the case of such concepts having dissimilar fragments of semantics in their descriptions.

4.2.1 Expanding the classification graph

As mentioned in the state of the art (refer to Section 2.2), intentional-based methods studying the structure of the definitions of concepts are hardly able to capture the seman-



(a) The sole computation of the semantic neighborhoods of A and B seems indicating a high similarity.

(b) Considering the semantic neighborhoods of C and D refines (by decreasing) the semantic similarity of A and B.

Figure 4.2 – Illustration of the impact of the semantics of a concept when computing similarity of other concepts.

tics that these concepts convey. To cope with this issue, some methods have introduced the use of LCS (or GCS) consisting of finding the best common subsumer of two concepts with regards to the common semantics that these concepts share. What we propose in this work, is grounded on these approaches and consists of generating the semantic neighborhood that is implied by a $\mathcal{SHOIQ}$-based concept. The idea behind generating the semantic neighborhood of each concept, is twofold. First, it allows to elicit the semantics conveyed by the concepts. Second, it allows to find what is the common semantics shared by two concepts and to get the best super-concept of the union of the semantics that both concepts do not share.

Unlike the other methods, we believe that generating the semantic neighborhood of all concepts must be done before starting any similarity computation. Indeed, we believe that computing the similarity of two concepts must be performed when (an only when) all possible semantics have been extracted from all defined concepts. In other words, we believe that the semantics of a concept C can impact the measure of similarity of two other concepts A and B. This idea is a generalization of what has been pointed out in [DSF08], except that it is applied on the semantics conveyed by each concept and not on the extensions of such concepts. It can be visualized in Figure 4.2. In our approach, the semantic neighborhood of any concept is built by using the subsumption relations that exist between restrictions, the hierarchy of roles as well as their transitivity. In addition, we also consider the subsumption axioms that exist between this concept and other defi-

nitions. Consider for instance, the following excerpt of the Wine [3] ontology:

$$
\begin{array}{lcl}
\text{CabernetSauvignon} & \equiv & \text{vin} \sqcap \\
 & & \exists \text{madeFromGrape}.\{\text{CSGrape}\} \sqcap \\
 & & \leq 1\ \text{madeFromGrape}.\top \\
\text{RedBurgundy} & \equiv & \text{Burgundy} \sqcap \text{RedWine} \\
\text{Burgundy} & \equiv & \text{vin} \sqcap \exists \text{locatedIn}.\{\text{BourgogneRegion}\} \\
\text{RedWine} & \equiv & \text{vin} \sqcap \exists \text{hasColor}.\{\text{Red}\} \\
\text{CabernetSauvignon} & \sqsubseteq & \forall \text{hasBody}.\{\text{Full, Medium}\} \\
\text{CabernetSauvignon} & \sqsubseteq & \exists \text{hasColor}.\{\text{Red}\} \\
\text{CabernetSauvignon} & \sqsubseteq & \forall \text{hasFlavor}.\{\text{Moderate, Strong}\} \\
\text{CabernetSauvignon} & \sqsubseteq & \exists \text{hasSugar}.\{\text{Dry}\} \\
\text{RedBurgundy} & \sqsubseteq & \exists \text{madeFromGrape}.\{\text{PinotNoirGrape}\} \\
\text{RedBurgundy} & \sqsubseteq & \leq 1\ \text{madeFromGrape}.\top \\
\text{Burgundy} & \sqsubseteq & \exists \text{hasSugar}.\{\text{Dry}\} \\
\text{vin} & \sqsubseteq & \geq 1\ \text{madeFromGrape}.\top \\
\text{vin} & \sqsubseteq & = 1\ \text{hasFlavor}.\top \\
\text{vin} & \sqsubseteq & = 1\ \text{hasSugar}.\top \\
\text{vin} & \sqsubseteq & \exists \text{locatedIn}.\text{Region} \\
\text{vin} & \sqsubseteq & = 1\ \text{hasColor}.\top \\
\text{vin} & \sqsubseteq & = 1\ \text{hasBody}.\top \\
\text{vin} & \sqsubseteq & \forall \text{hasMaker}.\text{Winery} \\
\text{vin} & \sqsubseteq & = 1\ \text{hasMaker}.\top \\
\text{madeFromGrape} & \sqsubseteq & \text{madeFromFruit} \\
\text{hasFlavor} & \sqsubseteq & \text{hasWineDescriptor} \\
\text{hasSugar} & \sqsubseteq & \text{hasWineDescriptor} \\
\text{hasColor} & \sqsubseteq & \text{hasWineDescriptor} \\
\text{hasBody} & \sqsubseteq & \text{hasWineDescriptor} \\
\multicolumn{3}{l}{\text{locatedIn}(\text{BourgogneRegion}, \text{FrenchRegion}) = \top} \\
\multicolumn{3}{l}{\text{locatedIn} \in N_{R^+} \text{ (i.e., locatedIn is transitive)}}
\end{array}
$$

The semantic neighborhood of *RedBurgundy* that we would like to establish, should consist of the semantics that this concept conveys, to further generate the pieces of semantics that are not represented in the ontology. In particular, this semantic neighborhood should include the disjunctive expressions that are semantically close to the definition of *RedBurgundy* (once written in Normal Form) such that:

$$\exists \text{locatedIn}.\{\text{BourgogneRegion}\} \sqcap \exists \text{hasColor}.\{\text{Red}\}$$

Then, the semantic neighborhood should include the DL expressions that can be learnt

3. The Wine ontology is a well-known ontology underlied by the DL $\mathcal{SHOIQ}(D)$, see `http://www.w3.org/TR/owl-guide/wine.rdf`

from the subsumption assertions declared in the ontology, e.g.

$$\exists \text{madeFromGrape.}\{\text{PinotNoirGrape}\}$$

or, from Burgundy description,

$$\exists \text{hasSugar.}\{\text{Dry}\}$$

The semantic neighborhood should also include the DL expressions derived from the hierarchy of roles, e.g.

$$\exists \text{hasWineDescriptor.}\{\text{Red}\}$$

resulting from the fact that hasWineDescriptor is a super-property of hasColor.
Finally, the semantic neighborhood of *RedBurgundy* should contain the DL expressions reflecting the transitiveness of some properties involved in its description, e.g.

$$\exists \text{locatedIn.}\{\text{FrenchRegion}\}$$

eliciting the relation between BurgundyRegion and FrenchRegion through the property locatedIn.
Towards this goal, we propose defining a family of generative functions that is able to build the semantic neighborhood of any concept written in Normal Form and underlied by a DL up to $\mathcal{SHOIQ}$.

Definition 4.1. Considering an ontology $\mathcal{O}$, with $R \in N_R$ a role, the generative function family $\mathcal{F}_\mathcal{G}$ consists of 6 functions defined from $\mathcal{W}_\mathcal{O} \times M$ to P, (with M and P subsets of ${\mathcal{W}_\mathcal{O}}^k, k \in \mathbb{N}$) as follows.

$$
\begin{array}{lcll}
 & \mathcal{W}_{\mathcal{O}} \times M & \rightarrow & P \\
\overline{\Phi}: & C, \{m_p\} & \rightarrow & \overline{\Phi_1}(C, \{m_p\}) \bigcup \overline{\Phi_2}(C, \{m_p\}) \\
\underline{\Phi}: & C, \{m_p\} & \rightarrow & \underline{\Phi_1}(C, \{m_p\}) \bigcup \underline{\Phi_2}(C, \{m_p\}) \\
\overline{\Phi_1}: & C \notin \{m_p\} & \rightarrow & \{C' \in \mathcal{O}/C \sqsubset C'\} \bigcup \\
 & & & \{X \in \overline{\Phi}(C', \{m_p\})/C \sqsubset C'\} \\
\overline{\Phi_2}: & \forall R.D \notin \{m_p\} & \rightarrow & \{\forall S.D, S \sqsubset R\} \bigcup \\
 & & & \{\forall S.X, S \sqsubseteq R, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \exists R.D \notin \{m_p\} & \rightarrow & \{\exists S.D, R \sqsubset S\} \bigcup \\
 & & & \{\exists S.X, R \sqsubseteq S, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \geq nR.D \notin \{m_p\} & \rightarrow & \{\geq nS.D, R \sqsubset S\} \bigcup \\
 & & & \{\geq nS.X, R \sqsubseteq S, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \leq nR.D \notin \{m_p\} & \rightarrow & \{\leq nS.D, S \sqsubset R\} \bigcup \\
 & & & \{\leq nS.X, S \sqsubseteq R, X \in \underline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \exists R.D \notin \{m_p\} & & \{\exists R.X, X \equiv \{b_1, \cdots, b_p\}, \\
 & R \in N_{R^+}, & \rightarrow & \forall \epsilon \in D, \exists \eta \in X \text{ such that} \\
 & D \text{ nominal} & & R.\{\epsilon\} \sqsubseteq R.\{\eta\} \\
 & \sqcap_{i=1}^{N} C_i \notin \{m_p\} & \rightarrow & \bigcup_{k=1}^{N} \{\sqcap_{l=1}^{k} C_{j_l}, \forall j_l \in [1,k] \wedge \\
 & & & 1 \leq j_1 < \cdots < j_l < \cdots < j_k\} \\
 & \text{otherwise} & \rightarrow & \emptyset \\
\underline{\Phi_1}: & C \notin \{m_p\} & \rightarrow & \{C' \in \mathcal{O}/C' \sqsubset C\} \bigcup \\
 & & & \{X \in \underline{\Phi}(C', \{m_p\})/C' \sqsubset C\} \\
\underline{\Phi_2}: & \forall R.D \notin \{m_p\} & \rightarrow & \{\forall S.D, R \sqsubset S\} \bigcup \\
 & & & \{\forall S.X, R \sqsubseteq S, X \in \underline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \exists R.D \notin \{m_p\} & \rightarrow & \{\exists S.D, S \sqsubset R\} \bigcup \\
 & & & \{\exists S.X, S \sqsubseteq R, X \in \underline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \geq nR.D \notin \{m_p\} & \rightarrow & \{\geq nS.D, S \sqsubset R\} \bigcup \\
 & & & \{\geq nS.X, S \sqsubseteq R, X \in \underline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \leq nR.D \notin \{m_p\} & \rightarrow & \{\leq nS.D, R \sqsubset S\} \\
 & & & \{\leq nS.X, R \sqsubseteq S, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\} \\
 & \sqcup_{i=1}^{N} C_i \notin \{m_p\} & \rightarrow & \bigcup_{k=1}^{N} \{\sqcup_{l=1}^{k} C_{j_l}, \forall j_l \in [1,k] \wedge \\
 & & & 1 \leq j_1 < \cdots < j_l < \cdots < j_k\} \\
 & \text{otherwise} & \rightarrow & \emptyset
\end{array}
$$

In this definition, P represents a set of elements generated by the functions of $\mathcal{F}_{\mathcal{G}}$ while M represents the concepts having been processed by any function of $\mathcal{F}_{\mathcal{G}}$.

The definitions of the generative functions take their foundations from subsumption properties whom almost are detailed in the Appendix A of this dissertation.
This means, for example, that $\overline{\Phi_2}$ applied to a concept $C \equiv \exists R.D$ generates all concepts $E \equiv \exists S.X$ where S is a super-property of $R (R \sqsubseteq S)$ and where X belongs to $\overline{\Phi}(D)$ which, as proved by Theorem 4.2 hereafter, is the set of the subsumers of D. In other words, applying $\overline{\Phi_2}$ on such a concept accounts for considering the subsumption properties P.3 and P.4 described in Appendix A. Now suppose that $C \equiv \exists R.D$ with D is a set of nominals ($D \equiv \{a_1, \cdots, a_n\}$). The application of $\overline{\Phi_2}$ on such a concept results in considering the subsumption property P.9 described in the Appendix A. Applying $\overline{\Phi_2}$ on $C \equiv \forall R.D$ accounts for considering subsumption properties P.1 and P.2 described in the Appendix. Applying $\overline{\Phi_2}$ on cardinal restrictions results in generating concepts that will further allow a semantic engine to take part of the properties P.5–8 when classifying the ontology. Finally, applying $\overline{\Phi_2}$ on a conjunction accounts for generating other conjunctions subsuming this one. In other words, for a concept $C \equiv A \sqcap B \sqcap D \sqcap E$, $\overline{\Phi_2}$ generates the concepts $A \sqcap B, A \sqcap D, \cdots, B \sqcap D \sqcap E$.
The same arguments can be applied to $\underline{\Phi_2}$.
Note that when implemented in an algorithm, the use of $\overline{\Phi_2}$ (resp. $\underline{\Phi_2}$) over intersections (unions) leads to an exponential increase of the number of time this same function is re-applied (for an intersection composed of 30 terms, $\overline{\Phi_2}$ generates around 2^{30} new intersections that must further be processed by the same function). As a consequence, we restricted its use in some of the experimentations that we drove (see Section 6.2.3) in particular when working with subsets of SNOMED-CT [4] containing concepts defined as a conjunction of 30 terms when written in their $\mathcal{SHOIQ}$ Normal Form.
Finally, this definition also comprises $\overline{\Phi_1}$ (resp. $\underline{\Phi_1}$) to take into account subsumers (subsumees) of any concept of the ontology, learnt from the assertions found in the ontology (e.g., if the ontology has GCIs, etc.).
These generative functions allow to build the semantic neighborhood of any concept, taking the form of a set of pseudo-concepts representing either subsumers (if the concept is a conjunction) or subsumees (the concept is a disjunction) of the concept. Thus, with the definition of $\mathcal{F}_\mathcal{G}$, we have the following theorems.

Theorem 4.2. *For any defined concept C in $\mathcal{O}$, $\overline{\Phi}(C)$ contains only subsumers of C.*

4. SNOMED-CT, Systematized Nomenclature of Medicine - Clinical Terms, `http://bioportal.bioontology.org/ontologies/SNOMEDCT`

Proof. Suppose that C_0 is a defined concept and that there is $\alpha \in \overline{\Phi}(C_0)$ such that $C_0 \not\sqsubseteq \alpha$. By using the notations $\mathcal{C}_1 = \{X \in \mathcal{O}/C_0 \sqsubset X\}, \mathcal{C}_2 = \{X \in \mathcal{O}/\forall Y \in \mathcal{C}_1, Y \sqsubset X\} \cdots$ and by noticing that it always exists a step N such that $\mathcal{C}_N = \emptyset$ (as the set of defined concepts is finite, it exists a concept [in fact $\top$] which is not subsumed by any other concepts) , we have $\overline{\Phi}(C_0) = \mathcal{C}_1 \cup \cdots \cup \mathcal{C}_{N-1} \cup \overline{\Phi_2}(C_N) \cup \cdots \cup \overline{\Phi_2}(C_0)$. By definition, any concept $Y \in \mathcal{C}_i, i = 1 \cdots N$ subsumes C_0. As a consequence, if such concept α exists it has to be in the subset $\overline{\Phi_2}(C_N) \cup \cdots \cup \overline{\Phi_2}(C_0)$.

Six cases happen then:

- C_i is a primitive concept, a set of nominals or a union of concepts. Hence, according to the definition of $\overline{\Phi_2}$, $\overline{\Phi_2}(C_i) = \emptyset$ and the proposition that α exists is false.
- C_i is an intersection of other concepts. In this case, however α cannot exist as $\overline{\Phi_2}$ only generates intersections that contain (and therefore subsume) C_i.
- $C_i \equiv \exists R.D$. As for any $S \in N_R$ such that $R \sqsubseteq S$, we have $\exists R.D \sqsubseteq \exists S.D$, then α exists if (and only if) it is in the subset $\{\exists S.X, R \sqsubseteq S, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)$. In other words, the existence of α depends on the existence of X, which exists only if D is a restriction, an intersection or a concept subsumed by other concepts.
- $C_i \equiv \geq nR.D$. As for any $S \in N_R$ such that $R \sqsubseteq S$, we have $\geq nR.D \sqsubseteq \geq nS.D$, then α exists if (and only if) it is in the subset $\{\geq nS.X, R \sqsubseteq S, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\}$. Again, the existence of α depends on the existence of X, which exists only if D is a restriction, an intersection or a concept subsumed by other concepts.
- $C_i \equiv \forall R.D$. As for any $S \in N_R$ such that $S \sqsubseteq R$, we have $\forall R.D \sqsubseteq \forall S.D$, then α exists if (and only if) it is in the subset $\{\forall S.X, S \sqsubseteq R, X \in \overline{\Phi}(D, \{m_p\} \sqcup C)\}$. Again, the existence of α depends on the existence of X, which exists only if D is a restriction, an intersection or a concept subsumed by other concepts.
- $C_i \equiv \leq nR.D$. As for any $S \in N_R$ such that $S \sqsubseteq R$, we have $\leq nR.D \sqsubseteq \leq nS.D$, then α exists if (and only if) it is in the subset $\{\leq nS.X, S \sqsubseteq R, Y \in \underline{\Phi}(D, \{m_p\} \sqcup C)\}$. Here, the existence of α depends on the existence of X, which exists only if D is a restriction, a union or a concept that subsumes other concepts.

Regarding the last four cases, the first part of the proof allows to say that if X is a concept or an intersection, then α cannot exist. By symmetry we also know that α cannot exist if X is a union of concepts onto which we apply $\underline{\Phi}$.

Finally, the only case to consider is X being a restriction. By recursion, we deduce then that such α exists if (and only if) $C_i \equiv \rho_1 R_1.(\rho_2 R_2. \cdots \rho_n(R_n. \cdots))$, with $\rho_j \in \{\forall, \exists, \geq, \leq\}$

and $n \to \infty$, which is not possible as the expression of any C_i is finite. As a consequence, α does not exist and the Theorem 4.2 is true. □

Theorem 4.3. *For any defined concept C in $\mathcal{O}$, $\underline{\Phi}(C)$ contains only subsumees of C.*

Proof. Using the symmetry between $\overline{\Phi}$ and $\underline{\Phi}$. □

Based on $\mathcal{F}_\mathcal{G}$, the algorithms written in the Appendix B can then be used to generate $\mathcal{PS}$, leading to Theorem 4.4.

Theorem 4.4. ***(Total Correctness).*** *For any concept $C \in \mathcal{O}$, the set of pseudo-concepts generated by the algorithms detailed in Appendix B contains either subsumers or subsumees of C. Moreover, the algorithm always terminates.*

Proof. As the algorithm is based on the definitions of $\mathcal{F}_\mathcal{G}$ and as any concept $C \in \mathcal{O}$ is in $\mathcal{SHOIQ}$ Normal Form, we know that for any concept $C \in \mathcal{O}$, the set of pseudo-concepts generated by the algorithm contains either subsumers or subsumees of C (according to Theorems 4.2 and 4.3).
Besides, for the same reasons than in the proof of Theorem 4.2, we know that $\overline{\Phi_1}$ terminates (for any concept C in an ontology $\mathcal{O}$, it exists a finite number of concepts defined in $\mathcal{O}$ that subsume C. By symmetry, we also deduce that $\underline{\Phi_1}$ terminates.
Because each concept C in $\mathcal{O}$ is a finite expression of terms, the number of nested restrictions in C is finite. Moreover, because we maintain a list of the concepts having been processed (through the set M), we avoid calling infinitely the different functions of $\mathcal{F}_\mathcal{G}$. For these reasons, $\overline{\Phi_2}$ (resp. $\underline{\Phi_2}$) terminates. As a consequence, SUBSUMERS, SUBSUMEES and the whole Algorithm 2 terminate. □

By applying the algorithm on each concept of $\mathcal{G}_\mathcal{O}$, pseudo-concepts can be extracted and further used by a classical semantic Web reasoner, to produce a refined classification graph $\mathcal{E}_\mathcal{O}$.

4.2.2 Computing similarity of concepts

Based on $\mathcal{E}_\mathcal{O}$, we can now propose a semantic similarity measure for any two concepts of $\mathcal{G}_\mathcal{O}$. Our idea is that this measure focuses on evaluating the common and the different semantics that these two concepts share. Towards this goal, and for any two concepts in $\mathcal{G}_\mathcal{O}$, we propose that their common semantics be based on the number of semantically

equivalent pseudo-concepts that they share in their descriptions. We also propose to determine a relevant common subsumer (RCS) in $\mathcal{E}_\mathcal{O}$ subsuming (at least) the union of the dissimilar semantics of these two concepts [5].

Determining the RCS of two concepts

In our approach, the RCS of two concepts accounts for the concept in $\mathcal{E}_\mathcal{O}$ realizing two conditions:

— The RCS must convey at least the union of the semantics of these two concepts and,

— Amongst the concepts that may realize the first condition, the RCS is the one with the most restrictive semantics (i.e., its semantics conveys as few concepts as possible, especially very generic concepts).

Towards this goal, we propose to associate a weight ω on each concept of $\mathcal{E}_\mathcal{O}$, using the number of its direct subsumers as well as the weight of its direct subsumees (if any). This weight represents the abstraction of a concept. In other words, the higher ω is, the more generic the concept.

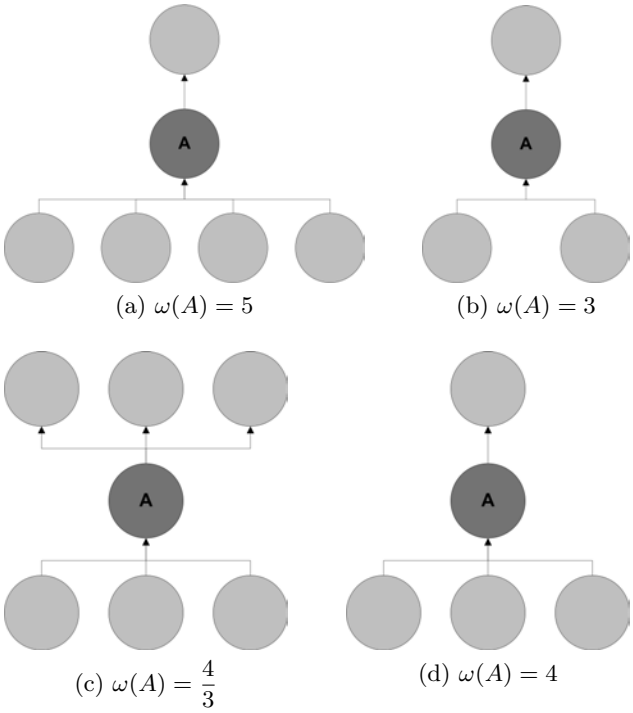
Definition 4.5. The weight of a concept C in $\mathcal{E}_\mathcal{O}$ is defined by the function ω such that:

$$\omega(C) = \frac{1 + \sum_{\alpha \in \underline{S_D}(C)} \omega(\alpha)}{\#\overline{S_D}(C)}.$$

Note that for the most abstract concept of the ontology (so without subsumers), the weight is set to $+\infty$.

The definition of ω has been motivated by some rules depicted in Figure 4.3. Figures 4.3a and 4.3b illustrate that the weight of a concept is function of its number of subsumees. The more subsumees a concept has, then the more abstract it is (and the higher is its weight). Figures 4.3c and 4.3d reflect our thought that a concept with more subsumers is one that matters more (i.e., it conveys important semantics w.r.t the domain which is modelled). As a consequence, such a concept must have a lower weight than one with few subsumers. Along with ω, we define a branch between two concepts of $\mathcal{E}_\mathcal{O}$ as follows.

5. Note that this method is computationally expensive and, although it is detailed in the following section, we did not applied it on very large ontologies (such as those used in Section 6.2.3). In the context of large ontologies the RCS simply accounts for the union of the dissimilar semantics of the two concepts compared (potentially leading to generating a new concept that was not in $\mathcal{E}_\mathcal{O}$, i.e. not computed through $\mathcal{F}_\mathcal{G}$).



(a) $\omega(A) = 5$ (b) $\omega(A) = 3$

(c) $\omega(A) = \frac{4}{3}$ (d) $\omega(A) = 4$

Figure 4.3 – The weight of a node is proportional to the weight of its subsumees and inversely proportional to its number of subsumers

Definition 4.6. A branch between two concepts E, F in $\mathcal{E}_\mathcal{O}$ and such that $E \sqsubseteq F$, is an ordered set of concepts that allows to go from E to F using the inclusion property. In this set, all the concepts are ordered by the inclusion property.

$$\mathcal{B}(E \to F) = \{C_i \in \mathcal{E}_\mathcal{O}, E \sqsubseteq C_1 \cdots \sqsubseteq C_i \sqsubseteq C_{i+1} \cdots \sqsubseteq F\}.$$

We further add to the definition of a branch that any concept C_i verifying the aforementioned inclusion property must be in the set of concepts (so in a branch). Moreover, because any concept in $\mathcal{E}_\mathcal{O}$ may have several direct subsumers (e.g. $E \sqsubseteq C_1$ and $E \sqsubseteq C_2$ with C_1, C_2 direct subsumers of E and $C_1 \not\sqsubseteq C_2, C_2 \not\sqsubseteq C_1$), several branches between two concepts of $\mathcal{E}_\mathcal{O}$ may exist. Accordingly, we define a branch set.

Definition 4.7. A branch set $\mathfrak{B}(E \to F)$ between two concepts E, F in $\mathcal{E}_\mathcal{O}$ and such that $E \sqsubseteq F$, contains all possible branches from E to F.

$$\mathfrak{B}(E \to F) = \{\mathcal{B}_i(E \to F), \mathcal{B}_i \text{ a branch between } E \text{ and } F\}.$$

Defining a branch allows to associate a weight giving an indication about how a concept is semantically close to one of its subsumers and therefore helps in determining the RCS of two concepts. Such weight is defined as follows.

Definition 4.8. The weight of a branch between two concepts E, F in $\mathcal{E}_\mathcal{O}$ is defined by the function Ω such that :

$$\Omega(\mathcal{B}(E \rightarrow F)) = \sum_{C_i \in \mathcal{B}(E \rightarrow F)} \omega(C_i).$$

The formula that we use to compute the weight of a branch takes into account subsumers, subsumees as well as siblings of a concept (as based on ω). Consider for instance the example displayed in Figure 4.4. In this example, two branches connect A to C. By computing the different weights, we derive that the best branch to consider is $\mathcal{B}_2$. Indeed, the subsumers that it contains are not generic (i.e. they do not contain a lot of concepts) and hence, even if they are more numerous than in $\mathcal{B}_1$, they are still better to consider.

$$\begin{aligned}
\mathfrak{B}(A \rightarrow C) &= \{\mathcal{B}_1(A \rightarrow C), \mathcal{B}_2(A \rightarrow C)\} \\
\mathcal{B}_1(A \rightarrow C) &= \{A, B_1, C\} \\
\mathcal{B}_2(A \rightarrow C) &= \{A, B_2, B_3, B_4, B_5, C\} \\
\Omega(\mathcal{B}_1(A \rightarrow C)) &= 59 \\
\Omega(\mathcal{B}_2(A \rightarrow C)) &= 58,5
\end{aligned}$$

Based on the aforementioned definitions, we consider that the RCS C_0 of two concepts E and F in $\mathcal{E}_\mathcal{O}$ is one of the direct subsumers of $E \sqcup F$ minimizing the sum of the weights of the two branches $\mathcal{B}(E \rightarrow C)$ and $\mathcal{B}(F \rightarrow C)$. In case of several concepts minimizing this sum, an arbitrary one is chosen.

Formally, $C_0 \in \underset{C \in \overline{S_D}(E \sqcup F)}{\arg\min} \Psi(C)$ with Ψ defined as follows:

$$\begin{aligned}
\Psi \quad : \quad \overline{S_D}(E \sqcup F) &\rightarrow \mathbf{R} \\
C &\rightarrow \min_{\mathcal{B}_i \in \mathfrak{B}(E \rightarrow C)} (\Omega(\mathcal{B}_i(E \rightarrow C))) + \min_{\mathcal{B}_i \in \mathfrak{B}(F \rightarrow C)} (\Omega(\mathcal{B}_i(F \rightarrow C)))
\end{aligned}$$

The semantic similarity formula

With the expanded graph and the RCS of any two concepts, we propose to define a semantic similarity measure respecting the following requirements.

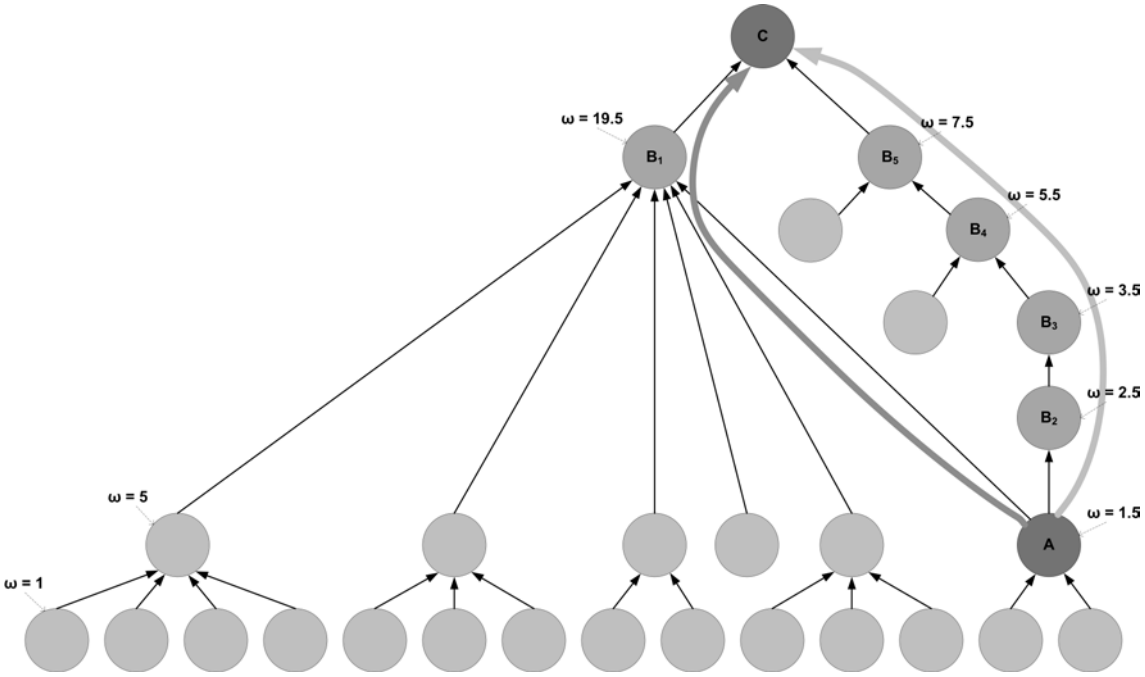


Figure 4.4 – Computing the weight of a branch. $\Omega(\mathcal{B}_1(A \rightarrow C)) = 59$; $\Omega(\mathcal{B}_2(A \rightarrow C)) = 58,5$.

— The similarity of two concepts is maximized if each of them share the same semantics. Specifically, suppose $C_1 \equiv D_1 \sqcap D_2$, $C_2 \equiv D_1 \sqcap D_2 \sqcap D_3$ and $C_3 \equiv D_1 \sqcap D_2$, then $sim(C_1, C_2) \leq sim(C_1, C_3)$.

— The similarity of two concepts must be symmetric. In other words, $sim(C_1, C_2) = sim(C_2, C_1)$.

— The similarity of two concepts is inversely proportional to the ratio of concepts subsumed by the two compared concepts to the concepts covered by their RCS. In other words, suppose that two concepts have different semantics (which is likely to happen), the more the RCS of their disjoint semantics is abstract w.r.t these concepts, the less similar the two concepts will be.

— the more concepts are similar, the more the value returned by our formula is close to 1. In other words, two concepts C, D are said to be completely similar if $sim(C, D) = 1$ and completely dissimilar if $sim(C, D) = 0$.

Based on these rules, the formula that we propose to apply is defined as follows.

$$sim(A,B) = \frac{2 \times n_c(A,B)}{n_t(A,B)} + \left[\left(1 - \frac{2 \times n_c(A,B)}{n_t(A,B)}\right) \times \left(\frac{2 \times IC(RCS(\delta(A,B), \delta(B,A)))}{IC(\delta(A,B)) + IC(\delta(B,A))}\right)\right].$$

In this formula, $n_c(A, B)$ accounts for the number of semantically equivalent concepts in $\overline{S_D}(A)$ and $\overline{S_D}(B)$. $n_t(A, B) = \#\overline{S_D}(A) + \#\overline{S_D}(B)$ and represents the number of direct subsumers of A and B. IC is the Information Content function that we define as follows:

$$IC(A) = -\log\left(\frac{1 + n_s(A)}{1 + n_s(\top)}\right),$$

where $n_s(A)$ is the total number of subsumees of A in the expanded graph $\mathcal{E}_\mathcal{O}$. Finally the function $\delta(A, B)$ is used to evaluate the different semantics of two concepts and is defined as follows:

$$\begin{array}{rccl} \delta: & \mathcal{G}_\mathcal{O} \times \mathcal{G}_\mathcal{O} & \rightarrow & \mathcal{W}_\mathcal{O} \\ & (A, B) & \rightarrow & \begin{cases} \sqcap_i C_i \text{ such that } C_i \in \overline{S_D}(A) \sqcap \neg\overline{S_D}(B) \\ A \text{ if } \forall C_i \in \overline{S_D}(A), C_i \in \overline{S_D}(B) \end{cases} \end{array}$$

This formula is inline with our requirements and indeed is symmetric (although δ is not a symmetric function, IC and RCS are symmetric), maximizes the similarity score of concepts sharing the same semantics (considering the left part of the formula), and takes into account their dissimilar semantics by computing their RCS (right part of the formula). If two concepts C, D share the same semantics and are determined by a reasoner as equivalent concepts, the value $n_c(C, D)$ will be half of $n_t(C, D)$. As a consequence, $sim(C, D)$ will be equal to 1. Finally, if two concepts C, D do not share any equivalent semantics ($n_c(C, D) = 0$), their similarity value will be equal to the right part of the formula. This value will be equal to zero if and only if $IC(RCS(C, D)) = 0$ (as both δ functions respectively return C and D) accounting for the RCS of C and D equals to $\top$. This case however, means that the two concepts share absolutely no common semantics in the expanded graph $\mathcal{E}_\mathcal{O}$ and is coherent with a similarity value determined at 0.

As our method does not rely on the use of extensions, it is able to recognize the semantic similarities of concepts asserted as disjoints (i.e. that will not share a common instance) and satisfies to the criterion of *disjointness incompatibility* formulated in [DSF08]. Because our method applies rewriting rules (it writes any concept of the ontology into its $\mathcal{SHOIQ}$ normal form) to expand the classification graph, it is able to elicit the underlying semantics of any concept and consequently can recognize when two different concepts are semantically equivalent. As a consequence, it satisfies the *soundness* and *equivalence soundness* criteria also formulated in [DSF08].

4.3 Example of application

Using the excerpt presented in Section 4.2.1 the semantic similarity of the concepts CabernetSauvignon and RedBurgundy would be computed as follows.
First, Algorithm 2 (see Appendix B) would compute the $\mathcal{SHOIQ}$ normal forms as follows.
For CabernetSauvignon:
Wine $\sqcap \exists$ madeFromGrape.$\{CSGrape\} \sqcap \leq 1$ madeFromGrape.$\top$.
For RedBurgundy:
Wine $\sqcap \exists$ hasColor.$\{Red\} \sqcap \exists$ locatedIn.$\{BourgogneRegion\}$.
Then, Algorithm 3 would generate their semantic neighborhood and return, for Cabernet-Sauvignon, the following:

$\forall$hasFlavor.{Moderate, Strong}
$\forall$hasBody.{Full,Medium}
$\exists$hasSugar.{Dry}
$\exists$hasColor.{Red}
$\exists$hasWineDescriptor.{Dry}
$\exists$hasWineDescriptor.{Red}
≤ 1 madeFromGrape.$\top$
$\exists$madeFromGrape.{CSGrape}
$\exists$madeFromFruit.{CSGrape}
$\exists$madeFromGrape.{CSGrape}$\sqcap \leq 1$ madeFromGrape.$\top$
Wine
$= 1$ hasBody.$\top$
$= 1$ hasColor.$\top$
$\exists$locatedIn.{Region}
PotableLiquid
$\forall$hasMaker.{Winery}
$= 1$ hasSugar.$\top$
≥ 1 madeFromGrape.$\top$
$= 1$ hasFlavor.$\top$
$= 1$ hasMaker.$\top$
≥ 1 madeFromFruit.$\top$
Wine$\sqcap \leq 1$ madeFromGrape.$\top$
Wine $\sqcap \exists$madeFromGrape.{CSGrape}

and for RedBurgundy:

$\exists$hasColor.{Red} $\sqcap$ $\exists$locatedIn.{BourgogneRegion}
Wine $\sqcap$ $\exists$locatedIn.{BourgogneRegion}
Wine $\sqcap$ $\exists$hasColor.{Red}
Wine
$\exists$locatedIn.{BourgogneRegion}
$\exists$hasColor.{Red}
$\exists$locatedIn.{FrenchRegion}
$\exists$hasWineDescriptor.{Red}
$\leq$ 1 madeFromGrape.$\top$
$\exists$madeFromGrape.{PinotNoirGrape}
$\exists$madeFromFruit.{PinotNoirGrape}
= 1 hasBody.$\top$
= 1 hasColor.$\top$
$\exists$locatedIn.{Region}
PotableLiquid
$\forall$hasMaker.{Winery}
= 1 hasSugar.$\top$
$\geq$ 1 madeFromGrape.$\top$
= 1 hasFlavor.$\top$
= 1 hasMaker.$\top$
$\geq$ 1 madeFromFruit.$\top$
$\exists$hasSugar.{Dry}
$\exists$hasWineDescriptor.{Dry}

Based on these semantic neighborhoods, a set of pseudo-concepts is generated which, together with all pseudo-concepts generated from other concepts defined in the Wine ontology, refines the direct subsumers of CabernetSauvignon and RedBurgundy concepts to the following.
For CabernetSauvignon:

DryRedWine
Wine $\sqcap$ $\exists$hasSugar.{Dry}$\sqcap$ $\leq$ 1 madeFromGrape.$\top$
$\forall$hasFlavor.{Moderate, Strong}
$\forall$madeFromGrape.{CSGrape, MerlotGrape}
$\forall$hasBody.{Full, Medium}
Wine $\sqcap$ $\exists$madeFromGrape.{CSGrape}
Wine $\sqcap$ $\forall$madeFromGrape.
{CFGrape, CSGrape, MalbecGrape,
MerlotGrape, PetiteVerdotGrape}

For RedBurgundy:

DryRedWine
Wine $\sqcap \exists$hasSugar.{Dry}$\sqcap \leq 1$ madeFromGrape.$\top$
PinotNoir
Burgundy

As a consequence, the following values are derived:

n_c(RedBurgundy, CabernetSauvignon) = 2

n_t(RedBurgundy, CabernetSauvignon) = 11.

δ(CabernetSauvignon,RedBurgundy) corresponds to the last five direct subsumers of CabernetSauvignon displayed above.

δ(RedBurgundy, CabernetSauvignon) corresponds to the last two direct subsumers of RedBurgundy displayed above.

A simulation taking into account all other concepts of the Wine ontology further returns the following values:

The RCS of both δ equals to RedWine.

$n_s(\top) = 266$.

$n_s(\delta$(CabernetSauvignon, RedBurgundy)) = 4.

$n_s(\delta$(RedBurgundy,CabernetSauvignon)) = 2.

$n_s(RCS) = 19$.

Finally, the semantic similarity of CabernetSauvignon and RedBurgundy that is obtained, is as follows:

$$sim(\text{RedBurgundy}, \text{CabernetSauvignon}) = \frac{4}{11} + \left[\frac{7}{11} \times \left(\frac{-2 \times \log(\frac{1+19}{1+266})}{-\log(\frac{1+2}{1+266}) - \log(\frac{1+4}{1+266})}\right)\right] = 0,75.$$

4.4 Conclusions

This chapter proposes that supporting nomadic users roaming in smart environments be realized by a searching process taking into account representation models defined in Chapter 3. Divided in two parts, this dissertation envisions that such a process may first be a succession of filters making use of the information provided by the representations of the connected devices, the involved user preferences, the location of both, etc. While the creation of such filters does not involve particular challenge, this chapter concentrates then on detailing a method to realize the second part of the process. In particular, this

method tries to provide answers to the two following challenges:

— Identifying similar connected devices (so that one may be used on behalf of an other, e.g. in failure recovery scenario) based on the semantics of their representation (at least the one not used in the filtering step of the searching process)

— Delivering a very accurate results set to an incoming user query, in case of a perfect match (provided by DL reasoners) may not be found

The method presented in this chapter addresses these two challenges by carefully designing a semantic similarity process taking into account a large portion of the semantics underlied by the DL $\mathcal{SHOIQ}$. Involving the creation of pseudo-concepts, this method is the cornerstone upon which other components providing user support can rely on (see Section 6.1 detailing the implementation of a system providing support to nomadic users).

Chapter 5

Distributing knowledge amongst smart environments

With the rise of the IoT and the plethora of devices now available from the Internet, supporting the nomadic users roaming across smart environments poses new challenges consisting of providing scalable processes able to associate devices with user or applicative requirements. This challenge is particularly emphasized by the fact that Semantic Web technologies – while they allow defining a unified representation to leverage the interoperability between devices, user and applicative requirements – heavily require time and process consumption to handle large collections of described entities, something that may likely to happen in wide smart environments. As a consequence, the benefits of using Semantic Web technologies are counter-balanced by the time and process that they require. Towards providing scalable search procedures, this chapter details the design of a distributed architecture consisting of federated semantic-enabled nodes. Each node of this architecture represents a delineated area of the smart environment (e.g., if the smart environment is composed of several rooms, a such area may be one of these rooms). A node further hosts and processes any semantic representation of devices, users and applications localized in the associated area. Finally, the solution proposes an interconnection scheme enabling any node to share what it has been able to compute (e.g., associations between a device and an application or a user) with other nodes managing neighboring areas.
This chapter takes back the idea published in [Chr12].

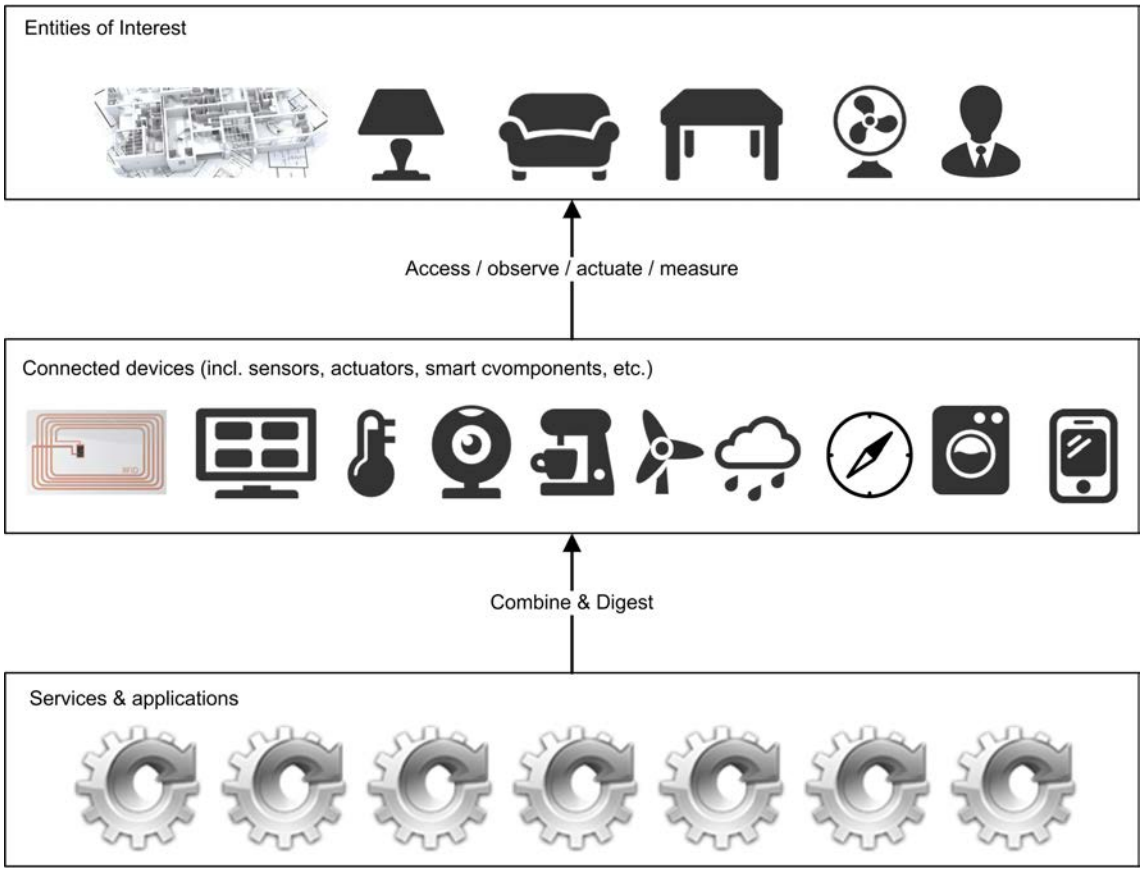


Figure 5.1 – IoT's three layers cake - Entities of Interest, connected devices and applications

5.1 Preamble

Aside pervasive and ubiquitous computing, the Internet of Things (IoT) is a paradigm that has recently gained momentum by proposing to extend the Internet to a variety of *things*, commonly sensed as physical entities of interest (EoI) to humans (e.g. a table, a room or even another human being). In this vision, these EoIs are represented through a set of properties that can be observed, measured or triggered through devices such as Radio Frequency IDentification (RFID) tags, sensors, actuators or other smart technological components (e.g. smartphones). These devices can further be coupled and orchestrated to provide digested information or complex services about things of interest to users (Figure 5.1).

While the IoT shares common goals with pervasive and ubiquitous computing – i.e. to enable that computing resources "weave themselves into the fabric of everyday life until they are indistinguishable from it" [Wei91] – this new paradigm brings additional challenges to

supporting nomadic users roaming in smart environments, in particular regarding to the complexity of selecting the right devices for a user or an application amongst the plethora that are now available.

Addressing this challenge poses indeed scalability issues, emphasized by the recent interest in providing sensors, actuators and connected devices through different platforms. As an example, the *Xively*[1] platform – offering a Web Application Programming Interface (API) to get information about sensors as well as to provide actuation capabilities – had an approximate pool of always reachable 115000 devices (as of March, 2012) distributed all over the world. *Nimbits*[2], *ThinkSpeak*[3] or *Thingworx*[4] are other examples of similar platforms. Considering that both users and connected devices may be mobile or may become temporarily unavailable, adds another layer of complexity as entailing to recompute such associations over the time.

While the Semantic Web technologies enable to create homogeneous, standardized and machine-processable representations (refer to Chapter 3), they however lack in providing efficient algorithms to reason over a large collection of descriptions and, depending on the DL underlying these descriptions, can turn to a computation nightmare if the whole set of descriptions has to be analyzed by a centralized Knowledge Base. To overcome scalability issues while still enabling accurate interoperability, we believe that the use of Semantic Web technologies must be thought of with deployment considerations in mind i.e. integrating the distributed and ubiquitous aspect of the IoT, in particular when involved EoIs and connected devices are dispatched in an indoor environment, prone to a high mobility. In this chapter, we propose then a distributed framework composed of nodes capable of processing Semantic Web descriptions and organized in a federated architecture. More precisely we propose that each node of the framework has local reasoning capabilities – i.e. be capable of processing semantic descriptions of connected devices (such that defined in Chapter 3) – and be able to cooperate with other nodes by exchanging the knowledge it has acquired about the devices, users and applications that it manages. We also propose that such knowledge be shared across geographically nearby nodes, considering location as a very important criterion when searching or associating users or applicative requirements

1. Xively – Public Cloud for the Internet of Things, `https://xively.com/`
2. Nimbits is a service you can use to record and share data on the cloud, `http://www.nimbits.com/index.html`
3. ThinkSpeak an open application platform designed to enable meaningful connections between things and people, `https://www.thingspeak.com/`
4. ThingWorx enables businesses to rapidly develop applications that connect people, systems, and the intelligent devices, `http://www.thingworx.com`

with connected devices (the likelihood that a device may be associated with someone is inversely proportional to their distance).
Our expectation is that incoming requests requiring associations will refer highly to a given location. Hence, we hope that having a geographically distributed management of knowledge about devices will enable an efficient discovery process through each node having gathered a sufficiently rich amount of knowledge about the devices living in (or nearby to) the geographical area it is bound to.

5.2 Federated architecture of nodes

In the literature, federated network systems refer to shared resources amongst multiple loosely coupled nodes [HM85] in order to optimise the use of those resources, improve the quality of network-based services, and/or reduce costs. Widely used in scenarios involving information sharing between different tiers [BBS04], such distributed systems can cope with storage and computation limitations and offer efficient – i.e. fast – search processes using optimization techniques [TOD05]. Due to these advantages, our view is that designing a federated system composed of semantic-enabled nodes may be a solution to speed up searching processes triggered in IoT-enabled Smart Environments, each node managing a limited set of connected devices, users and applicative requirements, but exchanging its knowledge with other peers to further enhance search results.
Supporting the aforementioned IoT paradigm through a federated system, is achieved by considering each loosely coupled node as the digital representation of a place hosting physical world objects. In the context of this dissertation, the concept of place is typically an indoor premise (e.g. a building, a room, etc.) as defined in Section 3.3. Note however that the idea developed in this chapter can be further readapted to address other kinds of places such as outdoor areas (e.g. a crossroad, a district, etc.). An example of node (say N) presented in this chapter may represent a meeting room equipped with a webcam, a presence sensor and other equipment. It may also be aware that some user (say U) with his set of applications (say A) is in this meeting room. Embedding storage and computing capabilities, each node manages then a pool of semantically described IoT stakeholders (referring to the Introduction of this dissertation) and can determine all possible associations between them (following our previous example, a node N computes and stores the representation of the webcam which is further associated with an application

from A, as matching an applicative requirement). Interconnecting these nodes allows a communication scheme where descriptions of IoT stakeholders as well as associations can be exchanged to maximize the aforementioned determination process of associations (e.g. the node N sharing semantic descriptions with another node M, so that M may prepare that some applications A of user U may be associated with another webcam). The following subsections describe the building blocks that would compose a node of such a federated system as well the interconnection scheme followed by these nodes.

5.2.1 Architecture of a node

Each node of a federated system has been designed to provide the following four capabilities:

— The storage and the processing of representations of users, applications and connected devices.

— The association process determining all possible interactions, e.g. realized by a SPARQL-DL engine if the representations are those detailed in Chapter 3.

— The propagation of aforementioned descriptions to other nodes in order to maximize the set of associations that such nodes will (re)compute.

— The capability to process user's queries and contact nearby nodes in case no results can be found. In particular, this step refers to the scenario presented in the Introduction and may be realized by the searching procedure as described in Chapter 4.

Figure 5.2 details the architecture of each node composing the federation. Although different implementations of such a node may be investigated, a possible embodiment can be a personal computer embedding computing and networking capabilities, and capable of handling a pool of processable resources.

In the context of this dissertation, the three kinds of resources that are managed by a node are the users, the applications and the connected devices. All these resources have a representation that can be processed and stored. While the node may try to associate applications with connected devices, user' queries may be handled differently. We recall that any considered resource can be mobile and therefore can enter or exit from a geographic place. We also assume the existence of a trigger process that notifies a node about such a join/exit event and provides it with the processable representation of the corresponding resource.

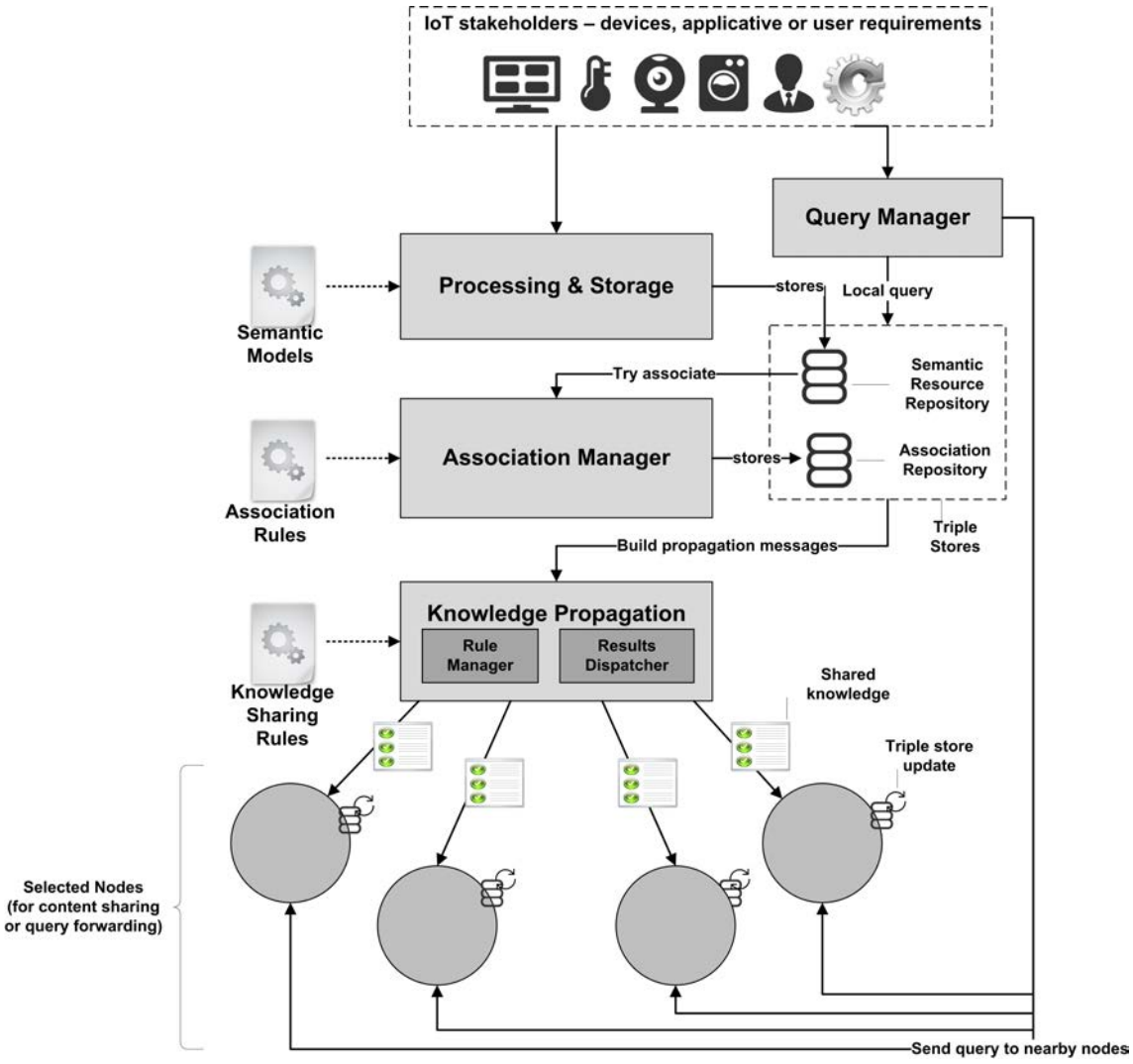


Figure 5.2 – Building blocks of a node

That being said, upon an incoming resource, the *Processing and Storage* functionality block of a node performs management functionalities including checking the validity of the representation of such resource. This check relies on semantic models such as those defined in Chapter 3. If compliant, the semantic description is translated to a set of RDF triples and inserted into the triple store of the node.

The stored semantic descriptions of the resources are then employed by the *Association Manager* that processes the requirements expressed by applications to find relevant connected devices i.e., providing the expected behavior. In the context of this dissertation, the association mechanism relies on using a SPARQL-DL engine [SP07] to process applicative requirements (we recall here that we have proposed a bijective function to translate a requirement in a SPARQL-DL query).

The *Knowledge Propagation* block uses Knowledge sharing rules defining the strategy of information sharing. Defined by a node manager (e.g. someone with administrative rights, managing the node by accessing to its configuration), examples of such rules can be the sharing of all representations of incoming devices, applicative requirements or users. Note however that in order to limit the generation of a high number of messages between nodes, trade-off such that restricting the sharing of information to the descriptions of incoming devices may be envisioned.

The *Knowledge Propagation* algorithm also uses the location model detailed in Section 3.3. Implemented by each node, the model enables to share information with nearby nodes (recall that a node is mapped to a geographic area). This location model allows localizing a place relatively to others (e.g. Chemistry lab is next to Computer Science lab) and serves as a basis to initialize and keep updated the federation system by defining how nodes are interconnected.

Finally, the *Query manager* processes incoming query sent by a user e.g., searching for some connected devices. In each node, finding results for a query involves two steps. First, a searching procedure such that the one described in Chapter 4 can be applied over the connected devices managed by the node. Benefiting from knowledge exchanged between nearby nodes, this first step allows finding corresponding devices that are in the vicinity of the user (either localized at the same place or in a nearby one). If no result is found, the request is forwarded to such nearby nodes that can themselves try to find relevant devices. As these nodes have other neighbors, the query is matched against a wider set of connected devices.

5.2.2 Interconnecting nodes and creating the federation system

To build a federated system composed of aforementioned nodes, we propose to create interconnections based on a 'container' approach, meaning that a place 'containing' other places results in as many interconnections as number of contained places (see for instance the curved arrows in Figure 5.3 interconnecting N_2 to N_4 and N_5 as a consequence of having the Chemistry lab and the Computer Science lab located in the 2^{nd} floor of a given building). In our vision, the place containing other places acts as a 'manager' of the places it 'contains'. The resulting federated system has moreover always a 'top-node' i.e. having no manager and representing the place that 'contains' all others (e.g., the University Building A in Figure 5.3).

Conceptually, a federation is represented as a directed acyclic graph (DAG) with no undirected cycles and where each non source node has an in-degree strictly equal to 1 and an out-degree above or equal to 0. The different places mapped to as many nodes in the federation are described using the model defined in Section 3.3 and as such, make use of the property 'contains' – also defined in this model, see Table 3.6 – to enable the aforementioned container approach.

By following this simple placement of rooms relatively to corridors, floors, etc. a federated system can be quickly deployed and extended, i.e. when a room is newly mapped to a node, such a node only needs to contact its 'manager' in order to declare itself as a new node of the federated system. This approach must however be used in conjunction with another process, enabling information acquired by a given node to be shared only with relevant nodes, i.e. those mapped to places nearby the place managed by the given node. As an example, Figure 5.3 presents the nodes of the Computer Science lab and the Chemistry lab as being interconnected to the node mapped to the 2^{nd} Floor of a University Building. However, it is not because both labs are in the 2^{nd} floor that they should exchange knowledge (consider for instance the case of a floor being 300 meters long, with both labs localized at the opposite corners. Exchanging knowledge may, in this case, be irrelevant as the distance separating both labs seems too high). By implementing the location model defined in Section 3.3, each node can be aware of all its 'neighbours' i.e. the ones it will share information with. This is made possible through a double cascading process (represented by black straight arrows in Figure 5.3) executed by each node when 'initializing' (recall that a node is a piece of software that is mapped to a place. Equipping a place with a node consists of starting this piece of software). Hence, at initialization,

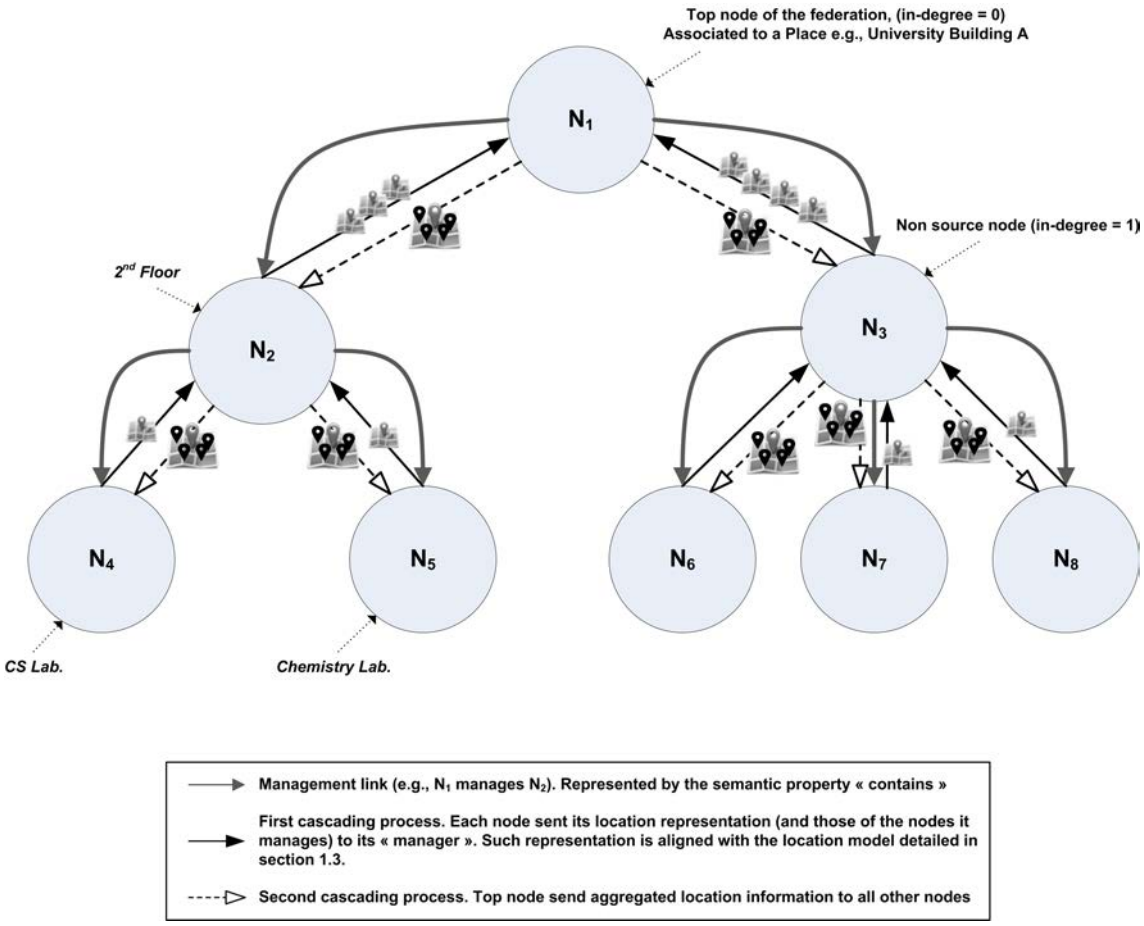


Figure 5.3 – Gathering overall nodes' location of a given federation network

each node communicates the description of the place it manages to the top node using a cascading process. The top node uses a semantic engine to merge this data from all nodes to obtain the overall distribution of nodes in the federation. The same cascading process is then used to relay this inferred distribution data to all nodes. When a new node (i.e. a place implementing some indoor location model concept and containing some connected objects) is added, the above cascading process is performed again. The new node can then begin sharing knowledge about the devices it manages.

5.3 Sharing knowledge between federated nodes

Our approach to share knowledge between nodes is applied on each node and can be divided in two steps. First, the process executes a set of rules against the aggregated location data, in order to select a set of nearby peers. Then, for each fact learnt by a node (e.g. new device or association detected), a message is created from this node to all selected recipients that update their respective Triple Stores dedicated to storing both semantic descriptions and such shared fact.

5.3.1 Customized Semantic Web rules

We define a set of processable rules using SWRL (recall Section 1.1.2) to enable facts discovered by one node to be shared with a subset of selected nodes, using customized SWRL built-ins triggering the initiation of notification messages. In addition, a learning process is coupled to these rules in order to attach a degree of confidence to the information that one node shares with the others. This point comes with the idea of anticipating the fact that some users or devices can move across different nodes. As an instance, assume that we have detected the following pattern in Ben's office: "Most people go by the coffee machine, then the corridor XY then the open offices AB". Knowing that Ben has moved from the coffee machine to the corridor XY may be sufficient to let the semantic node associated to the open offices AB knows that Ben is coming and may be interested in some connected devices "hosted" in the open offices, upon the reception of a message notifying that he has left the corridor XY. This would allow the node attached to the open offices to not replay its rule set after having been notified that Ben has entered the open offices. Although the set of rules is not finite and may be extended using the OWL import mechanism, this section details six particular rules (see Table 6.1 for their

expressions in SWRL) forming a basic strategy about the way a node could exchange knowledge with its peers. Such rules use the term *resources* to refer to described users, applications or connected devices.

— **Rule 1:** *When a resource has joined a place P, notify all the places accessible from P about such fact.*

— **Rule 2:** *When a resource has left a place P, notify all the places accessible from P that such resource could reach them.*

The next two rules replace the concept of accessibility in the first two rules with the concept of adjacency (a lower confidence score will then be computed, as adjacency does not imply direct accessibility).

— **Rule 3:** *When a resource has joined a place P, notify all the places adjacent to P about such fact.*

— **Rule 4:** *When a resource has left a place P, notify all the places adjacent to P that such resource may reach them.*

The final two rules take into account mobility of devices and EoIs by associating a learning process allowing nodes to notify other selected nodes that a device (resp. user) should join them in a near future. In detail, the fifth rule consists of notifying a place P_2 that a device (resp. user) may reach it soon. P_2 can then discover beforehand the associations between this device (resp. user) and the other resources it currently manages. As such associations are predicted, P_2 "locks" them (i.e. makes them unretrievable from searches) by tagging them as being "prepared". The sixth rule, finally, consists of unlocking these aforementioned associations by tagging them as being "available" (i.e. retrievable if searched).

— **Rule 5:** *When it has been learnt that any mobile resource always reaches a place P_2 after having reached P_1 and if a resource has just joined P_1, notify P_2 that such resource will join*

— **Rule 6:** *When the previous pattern has been learnt and that a resource leaves P_1, notify P_2 that a resource is joining*

The benefit of using SWRL rules to define how knowledge between nodes has to be exchanged is twofold.

First, it allows any *Place* owner to define additional rules, processable by a Semantic Web engine without requiring code to be developed (as long as the rules do not contain calls to customized built-ins unassociated with the engine). Thus, it allows policies to be associated to a strategy of knowledge sharing. As an instance, two different place managers may

decide two different strategies to share knowledge between nodes of the same federated network. Two different federated networks could also lead to different knowledge exchange models. Finally, different policies may be applied depending on their associated business models.
Second, use of SWRL allows built-ins to be developed and in particular, allows to link notification features to the "head" of a rule. Therefore, assuming a deployment where the different built-ins and models are known allows one to develop specific exchange protocols and rules.

5.3.2 Notification mechanism

Once having selected a set of peers with which to share some knowledge, a given node needs to send appropriate messages so that such peers will be notified of new content. Therefore, a notification mechanism needs to be implemented on each node and must be composed of a payload containing results to share and a header containing the appropriate route that a message has to follow to reach a previously selected peer.
Knowledge to share arises from the execution of aforementioned rules (Section 5.3.1) and is therefore a set of triples (the base unit in the Semantic Web).
Determining the path between a given node and the recipient of a message relies on the organizational aspect of the federation (recall Section 5.2 and Figure 5.3). Such path is exactly the list of nodes that need to be crossed, in order to find a "common manager" of both considered nodes.
Computing this path relies on the gathered and inferred location of all nodes and involves the anonymous property "inverse of givesAccessTo" (with *givesAccessTo* referring to the location model of Section 3.3 and its inverse provided by a Semantic Web engine). In more details, to establish a graph between two nodes A and B willing to share knowledge (considering a case were the knowledge has to be exchanged from A to B), we use the property *givesAccessTo* to build two subgraphs, respectively called *left subgraph* (starting with node A) and *right subgraph* (starting with node B). Building the left subgraph consists of asking a Semantic Web engine to provide all nodes $\{N_i\}$ such that "A *givesAccessTo* N_i" and to reiterate this request on the nodes having been found. The right subgraph uses the *inverse of*(*givesAccessTo*) property and therefore returns the list of nodes N_j such that "B inverseOf(*givesAccessTo*) N_j". Use of the property *givesAccessTo* allows one to find the ancestors of both the issuer and the recipient nodes. Hence, with this property, we build

two sub-graphs, one starting with the issuer and the other one starting with the recipient. Each time we found ancestors, we check if the two sub-graphs have a common node. If so, we merge them into a single graph over which we apply the Dijkstra [Dij71] algorithm which gives the shortest – and only – path between both nodes. Due to the particular nature of a federated infrastructure (recall that in Section 5.2 we said that the federation is a DAG with no undirected cycles), we are assured that the algorithm converges to one unique solution. Algorithm 1 details the building of these two subgraphs.

For a given result to share, the notification mechanism consists then of the generation of K messages (assuming K selected peers to notify). Each message contains a payload composed of a simple envelope to be routed properly as well as the result to share. Such envelope is composed of a list of nodes that need to be crossed. Accordingly, sharing a result consists of sending the message to the first node of the list and then goes through all other nodes appearing in the envelope. Upon receiving a message, a selected node processes it and updates its Triple Store. Figure 5.4 describes how the aforementioned components are used in a semantic node. From the semantic description of a resource (device or user), some triples are extracted (if joining the place) or retrieved (if leaving) from the Triple Store. Then, the triples feed the Association Manager that checks whether some associations can be created, updated or deleted. This leads to a list of resulting triples that are pushed into the Triple Store as updates. Finally, resulting triples also go to a Result Dispatcher that creates and conveys appropriate messages to a determined list of nodes.

5.4 Conclusions

To allow scalable search and management mechanisms for the IoT, this chapter details a distributed framework composed of nodes capable of processing Semantic Web descriptions and organized in a federated architecture. In addition to other approaches using Semantic Web technologies to enhance interoperability between devices or EoIs part of the IoT, our approach considers a particular deployment infrastructure where each node of such infrastructure is mapped to a physical environment (e.g. buildings, rooms, etc.). Considering such a set of nodes allows reasoning mechanisms local to each node to be setup and enables knowledge to be locally stored, avoiding a single and centralized repository to be flooded by queries (and data). This chapter also details a simple notification

Algorithm 1 Compute the left or right subgraphs SG of a given node n

```
// Create a DAG using JGraphT library
SG ← JgraphT.create_DAG(Node,DefaultEdge);

procedure CREATE_SUBGRAPH(n)(:)
    JGraphT.add_node(SG,n);
    analyze(n, direction);

end procedure
procedure ANALYZE(node, direction)(:)
    // Analyze node to build its subgraph SG
    subnodes ← [];
    predicate ← "";
    if direction = "left" then
        predicate ← "loc:giveAccessTo";
    else
        predicate ← "inverseOf(loc:givesAccessTo)";
    end if
    subnodes ← get_rdf_objects(node, predicate);
    if subnodes ≠ NULL and subnodes.length ≥ 1 then
        for all sn in subnodes do
            if sn ≠ NULL then
                add_node(sn, node);
                analyze(sn);
            end if
        end for
    end if

end procedure
procedure ADD_NODE(node, parent)(:)
    // Add a node in the DAG
    if node ∉ SG and parent ∈ SG then
        JGraphT.add_edge(SG, parent, node);
    end if

end procedure
procedure GET_RDF_OBJECTS(subject, predicate)(:)
    // Get a collection of objects object such as (subject, predicate, object) exist in the
knowledge base
    objects ← [];
    objects ← Reasoner.get_objects(subject, predicate);
return objects;
end procedure
```

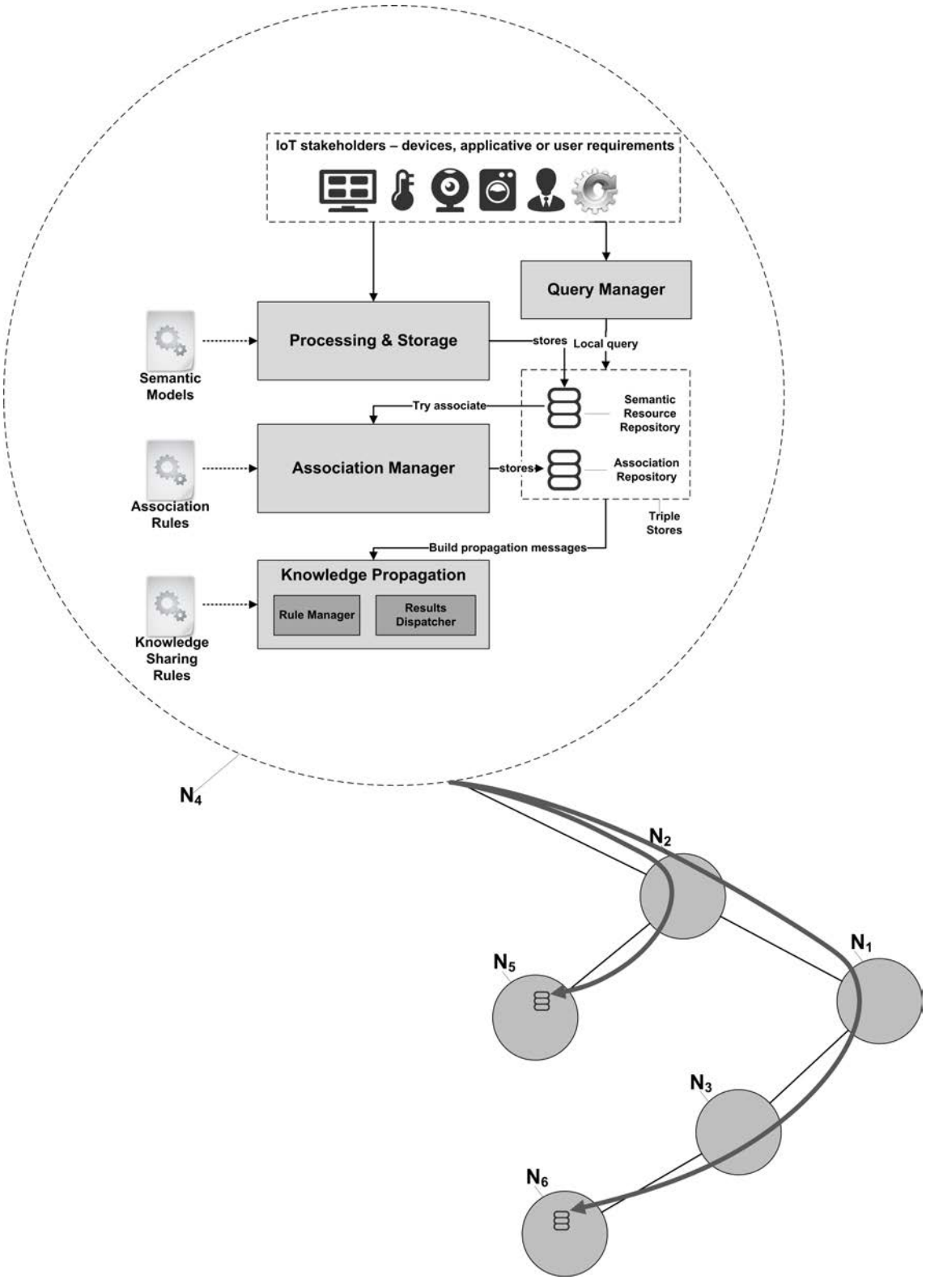


Figure 5.4 – Sharing knowledge between nearby nodes: Overall process

message mechanism, to enable nodes of the federation to share common interesting knowledge. Through this architecture, this chapter tries to keep the benefits provided by the Semantic Web (enhancing interoperability) without taking the drawbacks (reasoning tasks possibly computationally expensive).

Chapter 6

Experimentations

In this chapter we discuss experiments undertaken to validate the different ideas proposed in this dissertation.

These experimentations rely on the implementation of a *Virtual Resource Gateway* (VR Gateway), a framework able to host and process (descriptions of) connected devices and applications (referring to models detailed in Chapter 3) and to actuate upon them, based on requests received from users. One of the components of the VR Gateway handles search capabilities, implementing the semantic similarity measure presented in Chapter 4 as well as a SPARQL-DL engine to find connected devices matching applicative requirements. Jointly the implementation of a federated network of VR Gateways has been realized, enabling knowledge and search results to be shared between different VR Gateways (realizing the ideas exposed in Chapter 5). One of the components of the VR Gateway is therefore in charge of managing the communications entailed by the federated network. The first section of this chapter details all these implementations.

The different experiments having been driven are further presented and overall serve to validate our approach to support nomadic users roaming across smart environments. More specifically the experimentations assess the search capabilities that the components designed in the chapters 3 to 5 provide.

Analysis of these capabilities revolves around three experimentations, each of them addressing a specific focus.

The first experiment evaluates the ability of the whole implemented system to cluster similar representations. In particular, this experimentation assesses the semantic similarity measure (refer to Chapter 4) and if it is able to provide better results compared to

other methods existing in the literature. As this method does not rely on any particular applicative context (like the one of this dissertation referring to connected devices), the assessment of the measure is based on well-known datasets anybody can find on the Web. Thus, the results of this first experimentation allow to suppose that user queries or applicative requirements are likely to be better understood when sent to the system for searching results.

The second experiment evaluates the ability of the system to handle different types of queries, corresponding to what we think to be the typical applicative requirements sent from applications to configure. Consisting of assessing the mapping function defined in Section 3.4.3, this second experiment results in analyzing how the implemented system is capable of processing queries. In particular, the complexity of the semantics entailed by these queries is analyzed.

Finally the last experiment evaluates the relevance of the federated architecture and in particular checks the legitimacy of exchanging knowledge and search results amongst nearby nodes. We evaluate the scalability of this architecture compared to a centralized one, when performing search and we quantify the impact of the relocation of a connected device or an application on the federated architecture.

6.1 Implementations

The implementations described in the following subsections result from the different studies having led to several publications, such that the ones in [BCBT11, Chr12] and [CBTB12].

6.1.1 Exposing and processing connected devices and applications for users

To perform the different experiments, we have developed the VR Gateway, revolving around the realization of a framework adopting RESTful principles to expose connected devices and applications on the Web.

This framework accounts for representing a smart environment and relies on the concept of Virtual Resources, comprising the virtual representation of connected devices and applications. Virtual representation of devices represent the connected device, both to users through a digital representation (XHTML), and to other software components (other VRs)

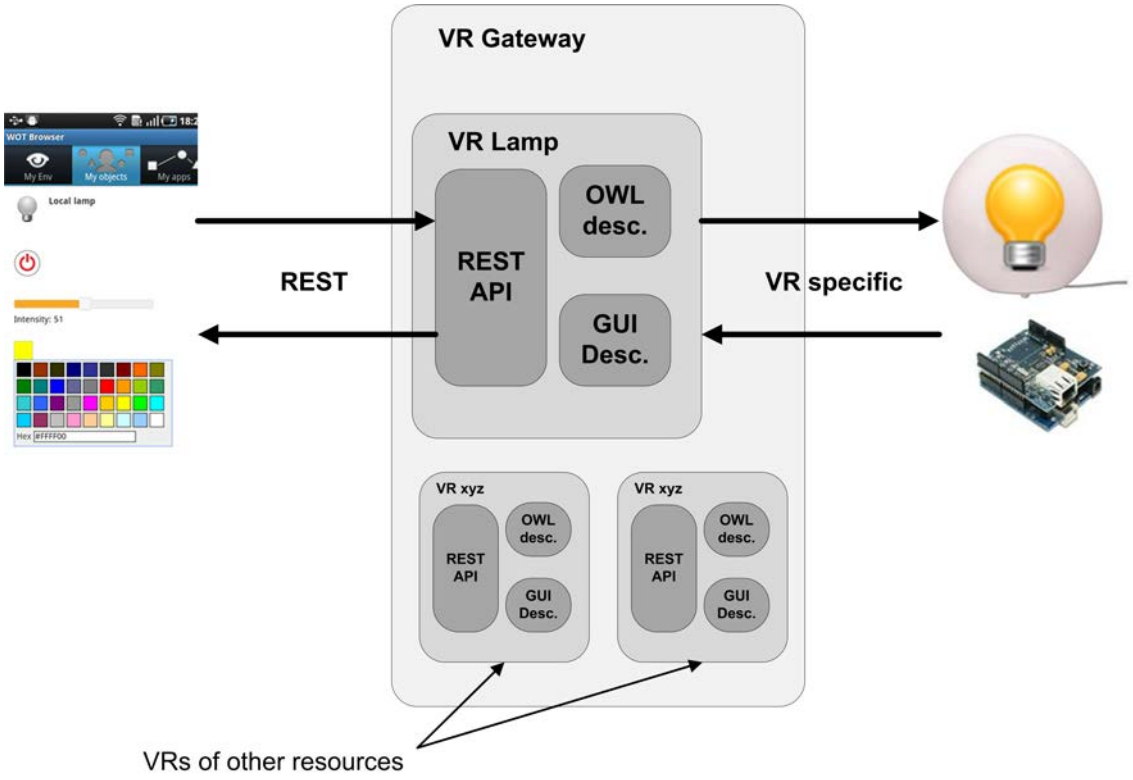


Figure 6.1 – VR Lamp Illustration

through a processable description and an exposed HTTP API (see Figure 6.1). For example, a connected lamp is represented through a Lamp VR exposing:

— REST APIs to enable the use of exposed functionalities and retrieve the XHTML representation.

— An OWL-based semantic description (aligned with the model presented in Chapter 3) to support reasoning on the composition or mash-up of this lamp with other VRs or applications.

— A user representation for visualization and interaction (described in XHTML), aiming to provide a single control point for interacting with resources of a smart environment, as well as enabling composition of these resources to create ambient intelligence.

In our implementation, considered applications accounts for XHTML “templates” i.e., containing a list of requirements to fulfill. For each application, this list of requirements is based on semantic annotations (RDFa) that are embedded in the template and that JavaScript controls – also embedded in this page – are able to retrieve and process. In particular, these controls translate a set of requirements to as many as necessary SPARQL-

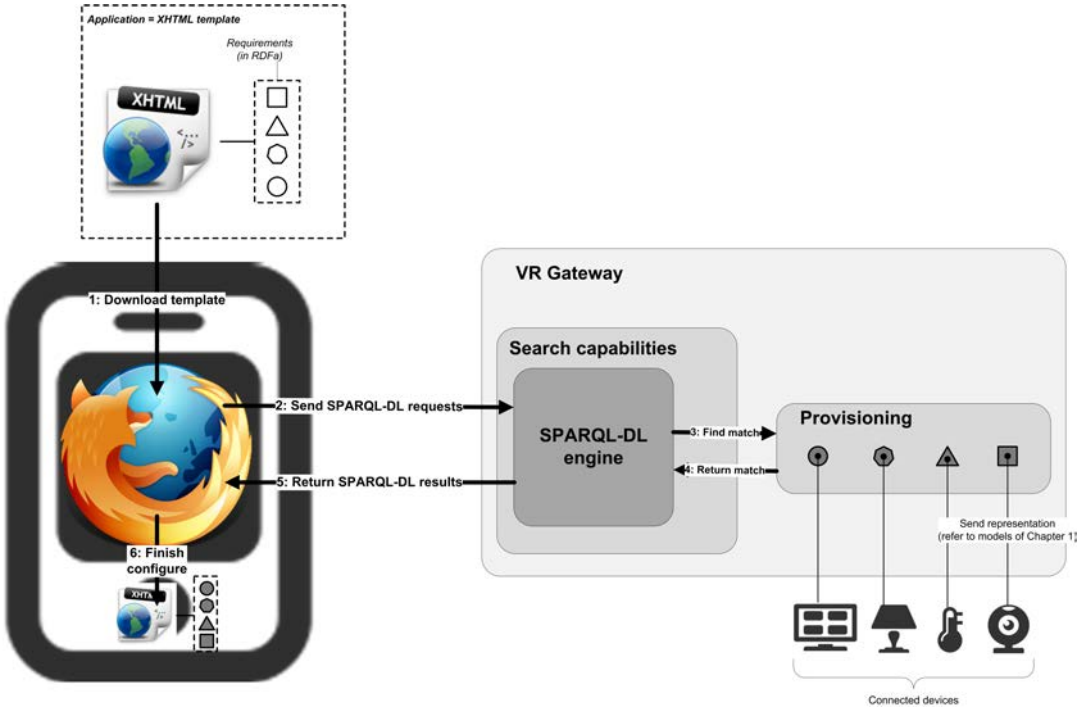


Figure 6.2 – Loading and configuring templates

DL requests (referring to Chapter 4) that are further sent to the VR Gateway which can finally process them and initiate searching mechanisms (see Figure 6.2). In terms of implementation, the VR Gateway is built on top of Equinox, an implementation of OSGi Framework R4.2 specifications [JMA10], and is therefore composed of OSGi bundles. We follow this approach because it provides an easy solution to enrich the set of functionalities and permits the implementation of VRs by third parties. The VR Gateway consists then to a hosting environment including VR implementations (i.e: HTML file, HTTP APIs and OWL description) of connected devices in the considered smart environment (e.g: lamp, phone, TV screen, mailbox). Thus, OSGi mechanisms allow dynamic instantiation of new VRs by adding required OSGi bundles at runtime, enabling then to handle dynamic reconfiguration of the smart environment i.e., supporting mobile devices and applications. The following describes the principal components of the VR Gateway, while Figure 6.3 displays the interactions that take place between the mobile phone of a nomadic user and the components of the VR Gateway.

VR Provisioning: The VR provisioning component manages the life-cycle of VRs and assigns them URLs. Since each VR must implement an OSGi service called *VirtualResource*, it is possible to catch a VR instantiation or deletion event using OSGi service discovery

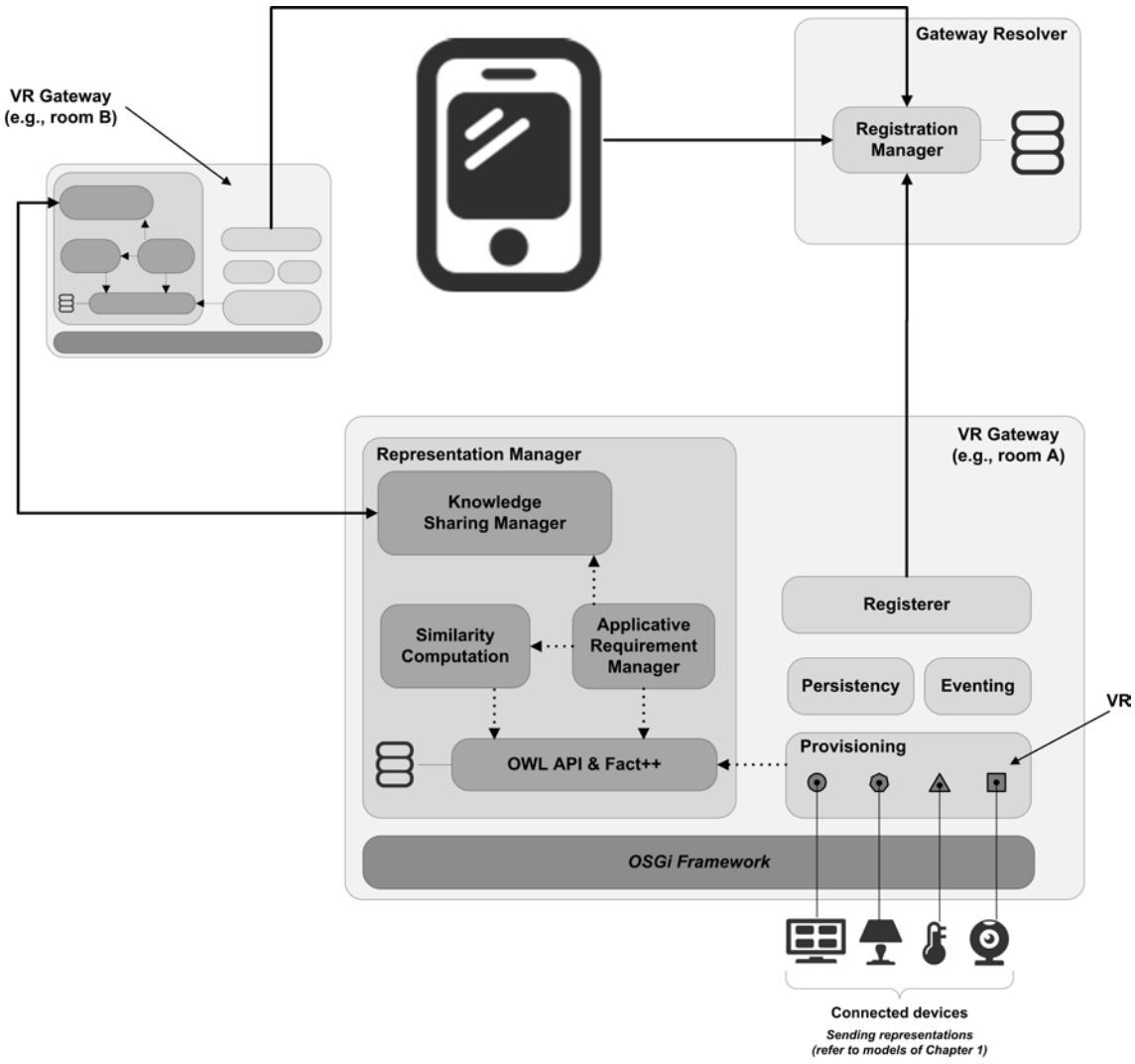


Figure 6.3 – Implemented Framework

mechanism, and thus keep up to date the list of VRs handled by the VR Gateway. The VR Provisioning element also exposes APIs that list instantiated VRs, instantiate new ones and access to their attribute values.

Persistency: The persistency component is used to automatically instantiate VRs when the gateway is starting. VR states are persisted into files so that they can be restored.

Eventing: The eventing component notifies clients of any changes on a VR. It defines the *NotificationEngineService* OSGi service which contains two features that post VR notifications and retrieve the event channel associated to a given VR. Each VR is responsible to disseminate its event channel through its 'eventchannel' attribute. The current implementation relies on Comet long poll mechanism [ME06], meaning that any client interested in changes on a dedicated VR should perform an HTTP GET on its event channel until a change occurs and resend it to catch the next one.

Representation Manager: This tool underlies the ideas presented in this dissertation (Chapter 3 to Chapter 5) by processing the semantic representations of connected devices and applicative requirements. In particular, one component of this tool performs similarity measurements (details in Section 6.1.2). Another one handles SPARQL-DL requests sent from applications or even users (details in Section 6.1.3). Finally, a last one initiates knowledge sharing and request forwarding (details in Section 6.1.4). Thus, managing the representation of devices and applications relies on Fact++, a DL-reasoner, as well as on the OWL API (providing Java constructs mapped to OWL concepts) allowing to process OWL descriptions.

Gateway Registerer: When the VR Gateway starts, the Gateway Registerer informs a component named the *Gateway Resolver*, acting as a repository maintaining the list of available VR Gateways. The Gateway Registerer also indicates to the Gateway Resolver its current location (referring to the location model described in Chapter 4). Afterwards, the Registerer sends periodical keep-alive requests. With the Gateway Resolver, any user willing to search for connected devices can send a query with his location (again, following model presented in Chapter 4). The Gateway resolver can then provide the URL of the relevant VR Gateway to contact.

6.1.2 Semantic similarity computation

Aligned with the developments made for the VR Gateway, the *Similarity Computation* block (see Figure 6.3) contains the implementation of the semantic similarity measure. De-

veloped in Java, this component makes use of a Java Native Interface (JNI) to further rely on the Fact++ DL reasoner for ontology management. As the semantic similarity method heavily requires the computation of branches between two concepts (recall Section 4.2.2), we setup a cache mechanism allowing to highly decrease the overall computation time to establish similarities between concepts (as an indication, the computation time decreased from 5 hours to 15 minutes by using this cache mechanism when testing our method on the Wine ontology). The component that we have released is articulated between seven different steps.

First, it loads an ontology where similarity computations have to be performed. In the context of this dissertation, this ontology would typically be a set of concepts refining the concept *Structure* defined in Chapter 3. Second it computes inferences, using the original ontology. The third step consists of rewriting any concept definition by replacing, when possible, any defined concept appearing in the left-hand side of the definition, by its expression. This step allows to detect cyclic definitions (e.g. $C \equiv \exists R.C$) and is mandatory to compute $\mathcal{SHOIQ}$ Normal Forms. The fourth step consists of replacing all defined concepts appearing in subsumption axioms (e.g. $C \sqsubseteq D$), by their expression computed in the third step. A consequence of this rewriting is that any subsumption axiom may become a GCI (as for instance a axioms such that $C \sqsubseteq D$ may be rewritten as the GCI $A \sqcap B \sqsubseteq E \sqcup F$). The fifth step consists of rewriting all concepts in their $\mathcal{SHOIQ}$ Normal Forms. The sixth step of our method consists of generating the pseudo-concepts of each concept originally defined in the ontology and having been rewritten in Normal Form. To achieve this operation, the default strategy having been implemented follows the algorithms detailed in Appendix B, i.e., is based on the definitions of $\overline{\Phi}$ and $\underline{\Phi}$. Once having generated the pseudo-concepts, we update the Knowledge Base by performing again a classification task.

The last step of our method consists of computing the similarity of concepts and in particular, in finding the RCS of any two fragments of dissimilar semantics.

6.1.3 Handling applicative requirements

Assessing the semantics defined in Section 3.4 entails the development of components enabling both the VR Gateway and a classical Web browser to present a list of matching devices to a user having downloaded an application template. While the former is im-

plemented in the VR Gateway under an *Applicative Requirement Manager* used together with the Fact++ DL reasoner; the latter is realized by a Javascript library complementing the XHTML description of an application template and capable of the following functionalities:

— Parsing semantic annotations (RDFa) of the XHTML description of the application template to retrieve requirements expressed as SPARQL-DL queries
— Sending these requests to the VR Gateway for processing
— Displaying the matching objects as a list to the user configuring the application template

Applicative Requirement Manager dynamics

Upon the reception of a SPARQL-DL query, the VR Gateway performs "local" deductions, involving the sole VRs monitored by this Gateway. This component is offered as an OSGI service embedded in the VR Gateway and relies on a SPARQL-DL library offered by Derivo [1]. This library is settled on top of the OWL API and extends the standard SPARQL specification with reasoning services that a semantic engine (in our case Fact++) provides. To not overwhelm the user with too many results an additional filter is set to each incoming SPARQL-DL query, limiting the number of returned results to at most ten per query. In case no results are found in the VR Gateway, the *Applicative Requirement Manager* relies on the sharing mechanism primarily built to set up the federated network of VR Gateways. Thus, instead of sharing a result this mechanism is asked to forward the request to nearby semantic nodes to help in finding relevant matches.

Javascript library

To initiate matching requirements with capabilities, a JavaScript library is linked to any application template description and executed by the Web browser of the user. When loading an application template, this library starts reading the semantic annotations and relies on the RDFQuery [2] library to extract the different SPARQL-DL queries. Then each request found is sent to the VR Gateway the user is associated with (recall that such association is made possible thanks to the Gateway Resolver). In particular, such SPARQL-DL queries are sent sequentially through HTTP POST requests (see Listing 6.1

1. Derivo SPARQL-DL API, `http://www.derivo.de/en/resources/sparql-dl-api.html`
2. RDFQuery, `https://code.google.com/p/rdfquery/`

representing a requirement used during our experimentation).

After having processed a request, the VR Gateway returns a JSON structure containing a set of matching VRs, which is evaluated by the library in order to display a list to the user configuring the application template.

Figure 6.4 summarizes the interactions between our JS library (considered as active when parsed by the Web browser) and the VR Gateway, upon the download of an application template.

```
POST <gateway_address>/sw/query HTTP/1.1                                    1
                                                                            2
                                                                            3
Content−Length:456                                                          4
PREFIX vr: <http://webofdevices.appspot.com/models/device.owl#>             5
PREFIX cap: <http://webofdevices.appspot.com/models/capability.owl#>        6
SELECT ?x WHERE                                                             7
{Type(?x,vr:Device),                                                        8
      PropertyValue(?x, vr:hasState, ?y),                                   9
      PropertyValue(?z, vr:isAccessibleFromState, ?y),                      10
      PropertyValue(?z, vr:realizes, ?c),                                   11
      PropertyValue(?c, cap:hasUserActionVerb, cap:EmitSignal)}             12
```

Listing 6.1 – SPARQL-DL query wrapped in an HTTP call

6.1.4 Creating the federation of semantic nodes

Handled by the *Knowledge Sharing Manager*, the implementation of a semantic node (and especially of all the components required to enabling intercommunications between different nodes, see Figure 5.2 of Chapter 5), relies on various Semantic Web technologies. To store and retrieve RDF descriptions of devices, applicative or user requirements, we have customized OWLDB [HKGB09] – a database backend for storing OWL triples – together with the OWL API [HB08]. In order to reason on these triples, we coupled them with Pellet[SPG+07], the only OWL-DL reasoner supporting the addition of customized SWRL built-ins, required to enable knowledge sharing process.

We used the *JGraphT*[3] open source library, providing features to build graphs as well as algorithms such as Dijkstra [Dij71] to find the path between two nearby nodes of a federated network. Establishing a path between two nodes A and B willing to share knowledge, is performed by feeding *JGraphT* with data retrieved from the aggregated and inferred loca-

3. JGraphT, a free Java graph library providing mathematical graph-theory objects and algorithms, `http://jgrapht.org/`

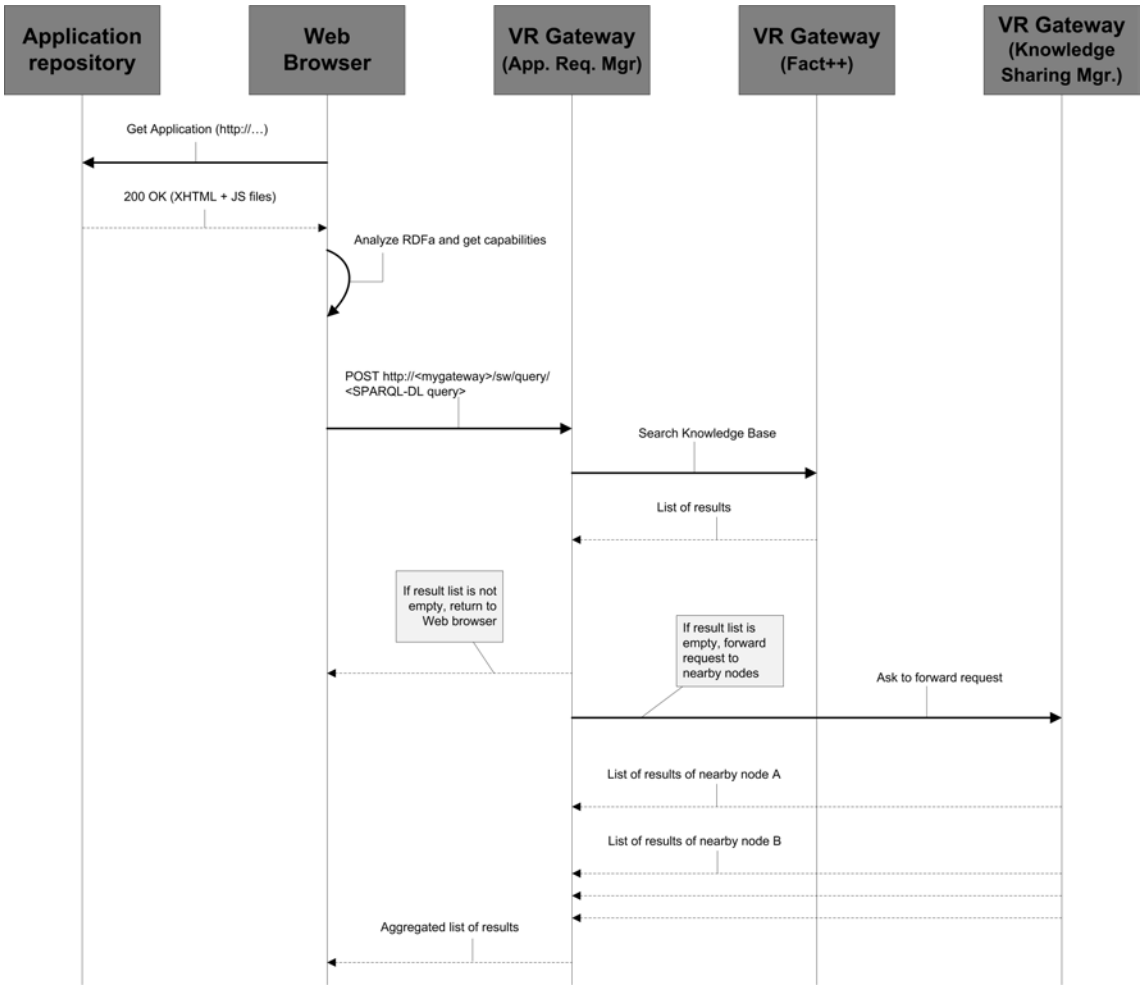


Figure 6.4 – Retrieving the VRs associated to connected devices complying with application template requirements

tion data (referring to the double cascading process detailed in Section 5.2.2). As already detailed in Section 5.3.2, considering the case where the knowledge has to be exchanged from A to B, we build two subgraphs with *JGraphT*, using the property *loc:givesAccessTo* – loc denoting the prefix used to refer to the location model of Section 3.3 – as well as its inverse *inverseOf*(*loc:givesAccessTo*). Implementation of the algorithm 1 is performed in Java and results in two subgraphs respectively called *left subgraph* (starting with node A) and *right subgraph* (starting with node B). Building the left subgraph consists of asking a Semantic Web engine to provide all nodes $\{N_i\}$ such as "A *loc:givesAccessTo* N_i" and to reiterate this request on the nodes having been found. The right subgraph uses the inverse of *loc:givesAccessTo* property and therefore returns the list of nodes N_j such that "B inverseOf(*loc:givesAccessTo*) N_j".

Rules mentioned in Section 5.3.1 have been written using SWRL. As an instance, Table 6.1 shows expressions corresponding respectively to rules 1 and 5 of the referenced section. These rules make use of prefixes referring to the indoor location model described in Section 3.3 (*loc* prefix), the representations defined in Section 3.2 (the *srv* prefix) as well as SWRL built-ins connected to machine learning processes (the *pattern* prefix) or notification mechanisms (*alert*, *notify* and *pnotify* patterns). They involve concepts, properties and constants that can be found on aforementioned models. About developed patterns, the features mentioned in these rules act as follows:

- *pattern:isNext* checks if the next node that a resource will join is a given node and returns a probabilistic score.
- *alert:notify* simply checks if a *resource* has joined or left a given node.
- *notif:notify* sends messages to nearby nodes about a fact that has (or will) happen(ed). Its associated probability score is equal to 1.
- *notif:pnotify* sends messages to nearby nodes about one fact that **may** happen with a certain probability. Getting such probability will not be described in this dissertation. Thus, the overall idea is to return a score taken into account the number of nodes that are *accessible from* or *adjacent to* a considered node.

To enable results to be shared between nodes, we rely on the SPARQL 1.1 specification that defines a dedicated section to update a graph with some triples – through "insert data" and "delete data" operations. The content of a notification message consists then of a SPARQL Update query containing the triple(s) to push to the Triple Store of the recipient of such information. Notification messages fired by SWRL built-ins are implemented

loc:Place(?$p1$) ∧ loc:Place(?$p2$) ∧ loc:givesAccessTo($p1$, ?$p2$) ∧
srv:IoTService(?s) ∧ alert:notify(?$p1$, ?s, loc:JOIN)
→ notif:notify(?$p2$, ?$p1$, ?s,loc:JOIN);

loc:Place(?$p1$) ∧ loc:Place(?$p2$) ∧
srv:IoTService(?s) ∧ srv:isMobile(?s, xsd:true) ∧
pattern:isNext(?$p1$, ?$p2$) ∧ alert:notify(?$p1$, ?s,loc:JOIN)
→ notif:pnotify(?$p2$, ?$p1$, ?s,loc:WILL_JOIN);

Table 6.1 – Rules 1 & 5 mentioned in Section 5.3.1

as HTTP messages containing one customized HTTP Request header (*X-nodes*) set up with the ordered list of nodes retrieved when establishing the path between two nodes. The message is sent to the first node to cross and then goes through all other nodes appearing in *X-nodes*. Each time the message is forwarded by a given node, its IP address appears in the standardized "*via*" header while it is removed from the *X-nodes* one.
As an example, Listing 6.2 contains a SPARQL Update query wrapped in an HTTP message sent from a node N_1 to a node N_6 (following the representation of Figure 5.3), to inform of an updated association involving one webcam and one person.

```
POST <N3_ip_address>/store/update HTTP/1.1
Content−Type : application/sparql−update
X−nodes : <N4_ip_address>, <N5_ip_address>, <N6_ip_address>
Via :
Content−length : 340
Prefix assoc: <http://models.iot−a.eu/association.owl>
DELETE DATA { <http://[N1_ip_address]/service/webcam1234.rdf> assoc:isAssociatedWith <http://dblp.l3s.de/d2r/
    resource/authors/Benoit_Christophe> } ;
INSERT DATA { http://[N1_ip_address]/service/webcam1234.rdf> assoc:isAssociatedWith <http://dblp.l3s.de/d2r/
    resource/authors/Suparna_De> }
```

Listing 6.2 – Message sent between two nodes

6.2 Experimentations

Sections 6.2.1 to 6.2.3 present various tests to assess the semantic similarity formula described in chapter 4, while Section 6.2.4 present tests assessing the semantics for applicative requirements. All these tests have been performed by using a standard Personal Computer equipped with a Pentium Core i5 processor and with 12GB of memory. The program is working on a 64 bits architecture with a maximum memory allocation setup at

Ontology	DL	# concepts	# roles
Wine	$\mathcal{SHOIQ}(\mathcal{D})$	133	17
FMA/NCI small	$\mathcal{ALCN}(\mathcal{D})$	3696	0
FMA whole ontology	$\mathcal{ALCN}(\mathcal{D})$	78988	0
NCI/FMA small	$\mathcal{ALCH}(\mathcal{D})$	6488	63
NCI/SNOMED small	$\mathcal{ALCH}(\mathcal{D})$	23958	82
NCI whole ontology	$\mathcal{ALCH}(\mathcal{D})$	66724	123

Table 6.2 – Some characteristics of the sets having been used

10GB. Note that as we are relying on other libraries, we can not optimize our program to make it run in parallel on each of the cores of our processor. Moreover, in order to assess the ideas presented in Chapter 5, Section 6.2.5 details an experimental setup consisting of 20 VR Gateways deployed on as many PCs, each PC being installed in a room or a corridor of a building (see also Figure 6.12). In this setup, each VR Gateway was exchanging HTTP messages – such as the one presented in Listing 6.2 – with its neighbors.

6.2.1 Towards the analysis of the semantic similarity measure

We assess our method on multiple sets available on the Web [4], the characteristics of which are displayed in Table 6.2. We use the Wine ontology to compare our approach with others found in the literature. In particular, we compare our method with approaches using the extensions of concepts. Results and discussion of this first experimentation can be found in Section 6.2.2. Then, we evaluate how our approach is able to process large datasets. Results of this second experimentation are reported in Section 6.2.3.

6.2.2 Comparison with the state of the art

In this first experimentation, we perform an evaluation of the results that our method produces compared to some methods of the state of the art. The goal of this first experimentation is twofold. First, it must enable to detect whether our approach seems to be a simple adaptation of some existing methods or if it really behaves differently when processing ontologies underlied with an expressive DL. Second, from the semantic similarity results being computed, this experimentation must allow to draw qualitative conclusions regarding to how our method handles complex definitions of concepts compared to the state of the art.

4. The Wine ontology, `http://www.w3.org/TR/owl-guide/wine.rdf`
The different subsets provided by the Ontology Alignment Evaluation Initiative, `http://oaei.ontologymatching.org`

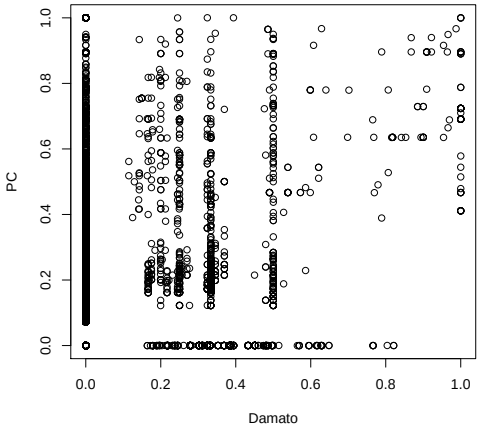
To this purpose, we use the Wine ontology and compare our approach to the methods presented in [dFE06, DSF08, LCM98, Lin98]. The results displayed in graphs 6.5a to 6.5d show that our method is not a simple transformation of the previous ones. Indeed, in these different graphs each point represents a pair of semantic similarity values between two concepts of the Wine ontology. The X-value of the point is what has been returned by one method of the state of the art while the Y-value has been computed with our approach. The absence of a curve in any of these graphs shows that our approach is not a transformation (linear, polynomial, exponential, etc.) of other existing approaches which, to some extent, tends to assess its novelty.

To drive the qualitative analysis, we select some pairs of concepts where our method presents different results from what is obtained by the state of the art (see an excerpt in Table 6.3).

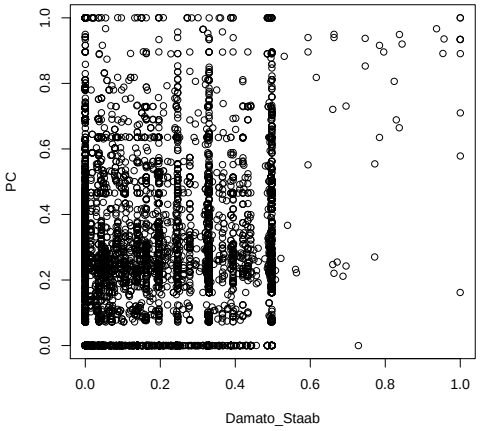
From this table, the first row emphasizes the limitation of the approach used in [LCM98] as it associates a high semantic similarity value to two concepts that do not share any semantics (other than the fact than both of them are subsumed by the very abstract concept "PotableLiquid", like all other beverages in this ontology). Indeed, this is due to the fact that the method proposed in [LCM98] only relies on path computation and that the small a path between two concepts is the higher their semantic similarity is. In the example of abstract concepts close to an even more abstract concept, the semantic similarity which is computed can be high, even if no common semantics is shared.

The second rows shows the limitation of the approach used by [Lin98]. Indeed, although "Alsatian Wine" and "Wine" share some semantics (an Alsatian Wine is a Wine which is in the Alsatian region), the method used in [Lin98] does not find any similarities because it relies on the instances of both wines. However, because the Wine ontology does not contain any instances of Alsatian wine, the result obtained by the approach in [Lin98] is null. Note that this problem clearly highlights the fact that most ontologies will never contain all possible instances of a given domain of interest and that consequently, adopting an approach that uses extensions of concepts may lead to inaccurate results.

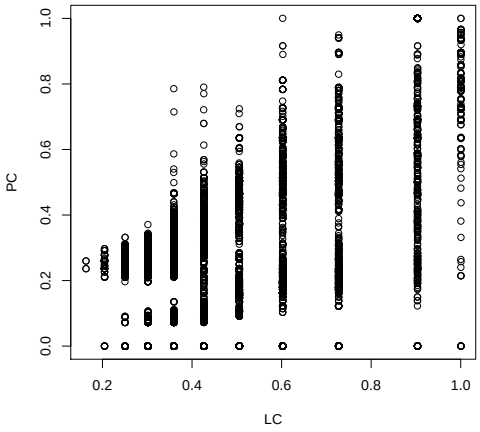
Finally, the last row shows the limitations of methods used in [dFE06] and [DSF08]. Indeed, although "RedBordeaux" and "Sauternes" wines share some semantics (a Sauternes Wine is a Bordeaux Wine and the Sauternes Region is located in the Bordeaux Region), the method used in [dFE06] is not able to return a score other than 0, because the fragments of common semantics do not share any common instances. The method used in



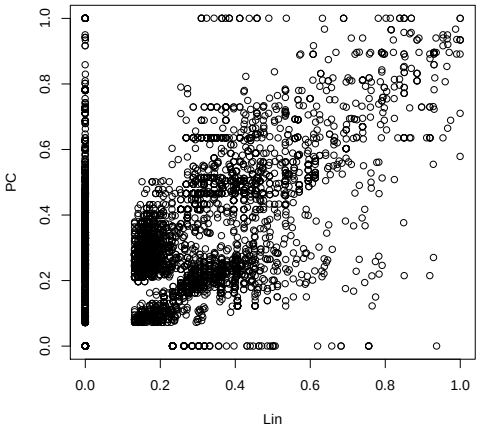
(a) X-axis: D'amato [dFE06]



(b) X-axis: D'amato [DSF08]



(c) X-axis: Leacock & Chodorow [LCM98]



(d) X-axis: Lin [Lin98]

Figure 6.5 – Semantic similarity computation performed on the Wine ontology. Each point represents a semantic similarity between two concepts. (X-value is computed using a method of the state of the art and Y-value corresponds to the value returned by our method)

Concept 1	Concept 2	D'amato [dFE06]	D'amato [DSF08]	L&C [LCM98]	Lin [Lin98]	Pseudo-concept
Juice	Sweet Riesling	0	0	0,6	0	0,23
Alsatian Wine	Wine	0,5	0	0,9	0	0,28
Red Bordeaux	Sauternes	0	0,25	0,73	0,82	0,79

Table 6.3 – Different results when computing the semantic similarity of some concepts in the Wine ontology

[DSF08] is able to recognize that the concepts have some similarities mainly because of the GCS that they compute. This method however, can not take benefits of the transitiveness of the property *isLocatedIn* as well as of the use of nominals in the definitions of these concepts. As a consequence, the value returned is not as high as the one found by our method that does consider transitiveness and use of nominals.

6.2.3 Test against large ontologies

Unlike the other approaches, our method requires a lot of computation in order to produce results, especially because of the three following steps:

— Rewriting all concepts of the ontology in their $\mathcal{SHOIQ}$ Normal Form (see Section 1.2).

— Deriving all pseudo-concepts using the algorithms in Appendix B, embodying the generative functions defined in Section 4.2.1.

— Computing the RCS of each pair of concepts, involving additional computation costs e.g., computation of branches (see Section 4.2.2).

Although a short execution time is not always a requirement when doing semantic similarity computation on concepts that do not evolve with time (in the context of this dissertation, measurements could primarily be performed on ontologies referring to structures defined by device providers. The similarity computation could then be performed by a *cron* process, e.g. everyday starting at 2am) we tried to investigate how our method was behaving with large ontologies. To this goal we run our approach on all the subsets presented in Table 6.2. During this second experimentation, we run into a number of problems that led to adapting our method with the three following changes. First, we realized that generating the $\mathcal{SHOIQ}$ Normal Forms of concepts was incredibly slow. We discovered that this was due to the current implementation of the OWL API that we were

using (use of a Java "TreeSet" structure that was trying to "order" elements composing a DL expression; this order being extremely long to compute). As a consequence, we decided to complement the OWL API with additional functionalities using a different structure (i.e., a "HashSet" where order does not matter).

Second, we realized that the computation of pseudo-concepts for concepts expressed through a long conjunctive form was leading to a huge computation time. Indeed, consider for instance a concept $D \equiv C_1 \sqcap \ldots \sqcap C_n$. Obtaining all these pseudo-concepts using the definition of $\overline{\Phi}$ accounts for generating $2^n - 2 - n$ intersections ($C_1 \sqcap C_2, \cdots, C_{n-1} \sqcap C_n, \cdots, C_2 \sqcap \cdots \sqcap C_{n-1}$) and re-applying $\overline{\Phi}$ and $\underline{\Phi}$ to each of them. In particular in the different SNOMED-CT subsets used in the second experimentation, some concepts were equal to a conjunctive form composed of 30 elements (hence $2^{30} - 32$ intersections to generate). To cope with this issue, we implemented a second strategy allowing the user to select the maximum number of intersections to build, by positioning a parameter giving the maximum length of an intersection that our algorithm can create. With this parameter, we voluntarily downgraded our method but ensured that large ontologies could possibly be processed (in an acceptable time).

Third, we realized that computing the RCS of the dissimilar semantics happening in the descriptions of any two concepts defined in a large ontology was not scalable. To address this problem, we then decided to set the RCS as accounting to the union of the two dissimilar fragments of semantics. As an instance, for $C \equiv A_0 \sqcap A_1 \sqcap A_2$ and $D \equiv A_0 \sqcap A_3 \sqcap A_4$, the RCS$(A_1 \sqcap A_2, A_3 \sqcap A_4)$ accounts for a potentially new concept $E \equiv (A_1 \sqcap A_2) \sqcup (A_3 \sqcap A_4)$ [5]. Note that because these subsets do not contain individuals, we did not try to compare our method with [dFE06, DSF08] and [Lin98] as such methods were not applicable. We however compared our approach to the one in [LCM98] and obtained the same kind of results than those displayed in Figure 6.5c (consistent with what was said in the previous section as the method proposed by Leacock & Chodorow is not able to provide accurate measurements for ontologies underlied by a DL, whatever its expressivity).

Figures 6.6 to 6.8 give indications on how the time increases with the complexity and the size of the datasets when respectively:

— computing the $\mathcal{SHOIQ}$ Normal Form of the concepts,

— generating the pseudo-concepts and,

— computing semantic similarity measurements (the graph displayed in Figure 6.8

5. In this context, the RCS accounts then for the LCS, as defined by [CBH92]

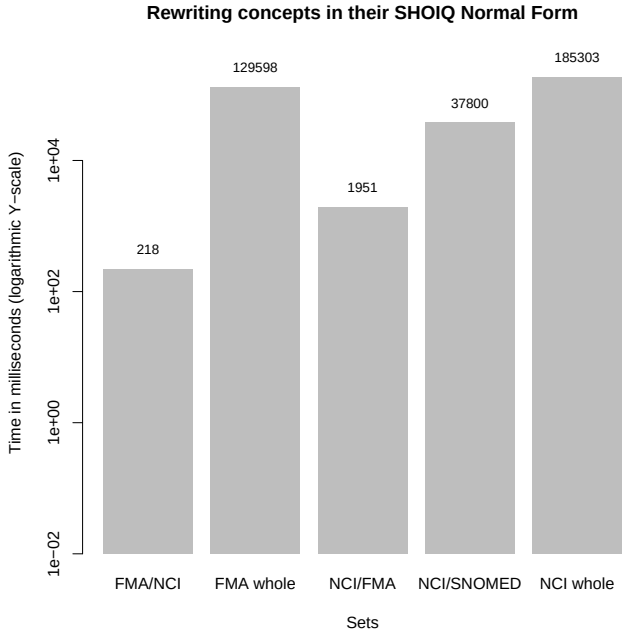


Figure 6.6 – Rewriting concepts in their $\mathcal{SHOIQ}$ Normal Form.

represents the average time to compute one single semantic similarity measure between two concepts).

Although these figures do not intend to give a generalization of the behavior of our approach, they allow to visualize that our approach can be computationally expensive (e.g. Figure 6.7 shows that it takes more than a full day to compute the entire set of pseudo-concepts for the subset "NCI whole ontology"). However, they also show that by doing concessions in regards to our original approach, large datasets can be processed while the similarity measurements being found are still much better than the methods known in the state of the art. Finally, Figures 6.9a to 6.9e represent the different steps of our process applied to the various sets represented in Table 6.2 (all except the Wine ontology). These steps are recalled hereafter:

1. Loading the ontology.
2. Computing inferences of the original datasets.
3. Detecting cycles in concept definitions.
4. Rewriting subsumption axioms.
5. Rewriting concepts in their $\mathcal{SHOIQ}$ Normal Form.
6. Computing pseudo-concepts.

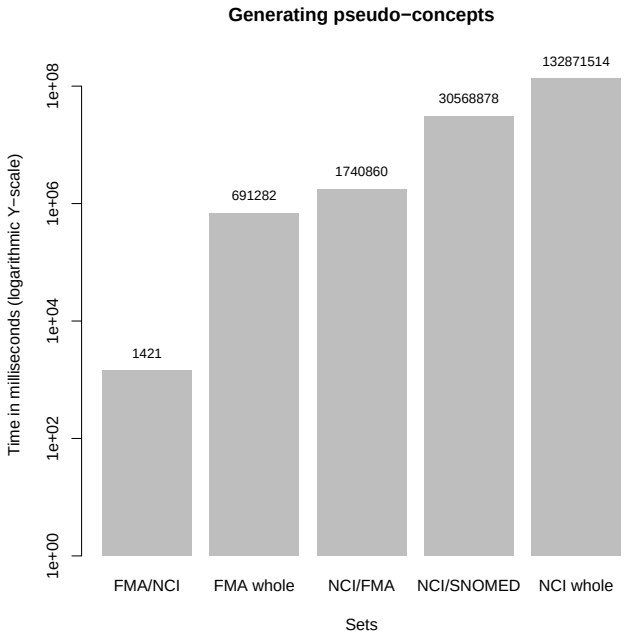


Figure 6.7 – Generating Pseudo-concepts.

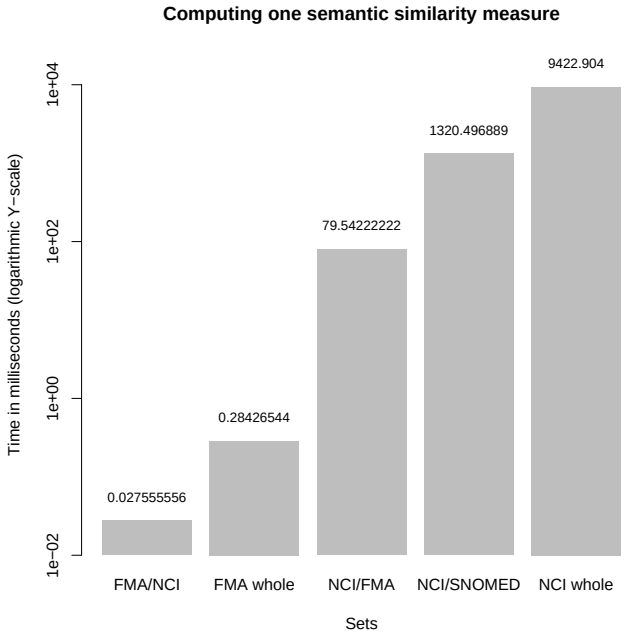


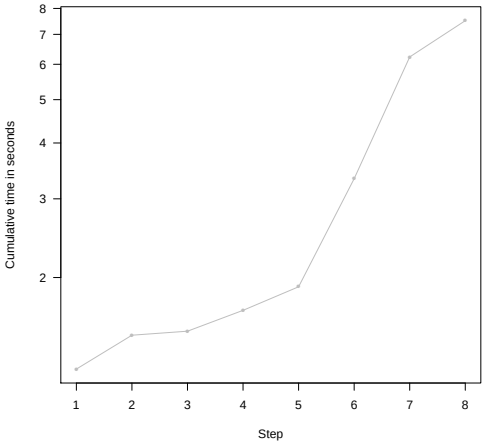
Figure 6.8 – Computing one semantic similarity measure.

7. Recomputing inferences with new concepts.
8. Computing 1000 similarity measurements.

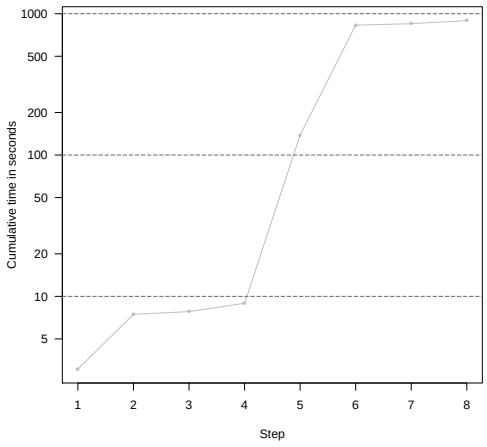
In these figures, the generation of pseudo-concepts clearly appears as the step that requires the most computation time.

6.2.4 Assessing the support for applicative requirements

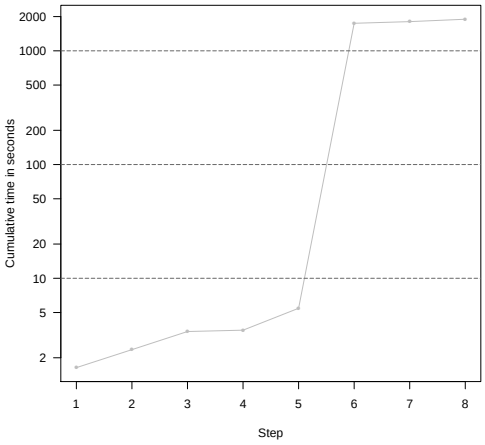
In this experimentation, we essentially look at the time and memory required by the VR Gateway to find connected devices matching applicative requirements. In this context, the experimentation is performed by considering 10 requirements (Table 6.4) of various complexities focusing on finding connected devices either generating a specific content (referring to the concept Structure defined in Section 3.2) or realizing some capabilities (referring to the Capability concept defined in Section 3.4.1).
Each requirement is sent to the VR Gateway having pre-loaded a set of instances of respectively connected devices, capabilities and contents beforehand. To test if the size of data is impacting the time taken by the system to analyze requirements, various sets containing either little or high numbers of connected devices, capabilities and contents are generated (see Table 6.5). For each of these sets, the experimentation measures the memory and time required by the *Applicative Requirement Manager* (and indirectly by the Fact++ DL reasoner) to load and process it. Results show that all sets are processed using less than 500MB of memory, fostering the idea that smart environments can be created at a little cost, simply by using a Personal Computer to host a VR Gateway. Thus, this test reports that the time spent to process representations tends to follow an exponential function based on the number of RDF triples (recall that an RDF triple is the base unit enabling to express OWL statements), and emphasizes the use of a federated network to manage smart environment with a huge number of connected devices. Table 6.5 summarizes the measures obtained during this test. The second test of this experimentation consists of measuring the time taken to answer to applicative requirements. Reported in the two Figures 6.10 and 6.11, this test reveals that the complexity of a SPARQL-DL query, the absence of constraints in the query and the size of the dataset are three factors affecting the computation time required to handle an applicative requirement.
In particular, in both figures it is shown that Requirement 4 is processing time-guzzler compared to Requirements 5 and 6, while the only difference between them is the addition of constraints (Requirement 4 is less constrained than Requirement 5 itself less constrained



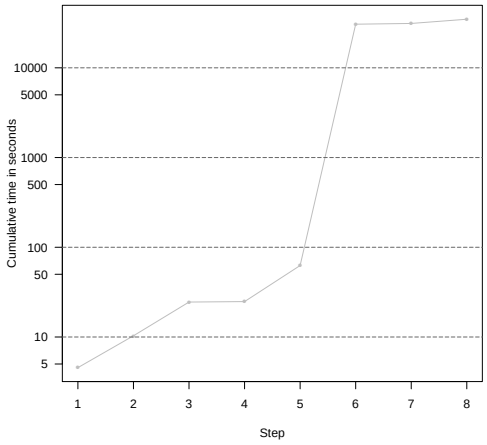
(a) FMA small overlap NCI



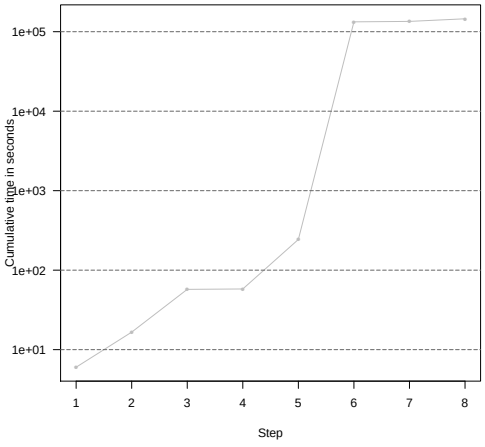
(b) FMA whole ontology



(c) NCI small overlap FMA



(d) NCI small overlap SNOMED



(e) NCI whole

Figure 6.9 – Cumulative time of our process applied on large ontologies.

Req. 1.	Get all VOs known by the semantic framework
Req. 2.	Get all functionalities of all known VOs
Req. 3.	VOs realizing a given Capability (knowing they were no solution)
Req. 4.	VOs realizing a Capability having a given UserActionVerb
Req. 5.	Complementing Req. 4 by adding a constraint on Target
Req. 6.	Complementing Req. 5 by adding a constraint on PerceptionModality
Req. 7.	VOs realizing either a Capability A or a Capability B (testing disjunctions)
Req. 8.	VOs generating a kind of content (knowing they were no solution)
Req. 9.	VOs generating a kind of content
Req. 10.	VOs generating a kind of content (say "A") or a content made of this "A" (testing disjunctions and use of a transitive property)

Table 6.4 – The 10 requirements used in the 2^{nd} experimentation

Set	VO	Capabilities	Content	Max memory peak (MB)	Processing time (sec.)
1	5	25	35	20	4,5
2	20	100	140	17	6,0
3	50	250	350	28	8,3
4	100	500	700	95	23,4
5	200	1000	1400	275	35,8
6	500	2500	3500	457	93,3
7	1000	5000	7000	470	237,3

Table 6.5 – Datasets used in the 2^{nd} experimentation

than Requirement 6).

Both figures also highlight that Requirement 10 requires much more time than Requirement 9, showing that the use of transitive properties or disjunctions may lead to a query which is computationally expensive to process. In other words, this test gives indication about the fact that our method, although capable of processing expressive requirements is relatively slow. This conclusion is emphasized by Figure 6.11 that shows that the time required to handle applicative requirements grows up exponentially with the size of the connected devices managed by the VR Gateway. In particular, Requirements 4 and 10 are already slow to process for a relatively little number of connected devices (50). While this may sound problematic in the context of Internet of Things – where billions of devices may become connected, it however emphasizes the need for a distributed management approach, where queries are checked against a limited set of connected devices and further sent to other management systems in case of no answer can be provided. The result of this experimentation explain why this dissertation further considers the idea of creating a federated network of semantic nodes to deliver scalable search capabilities.

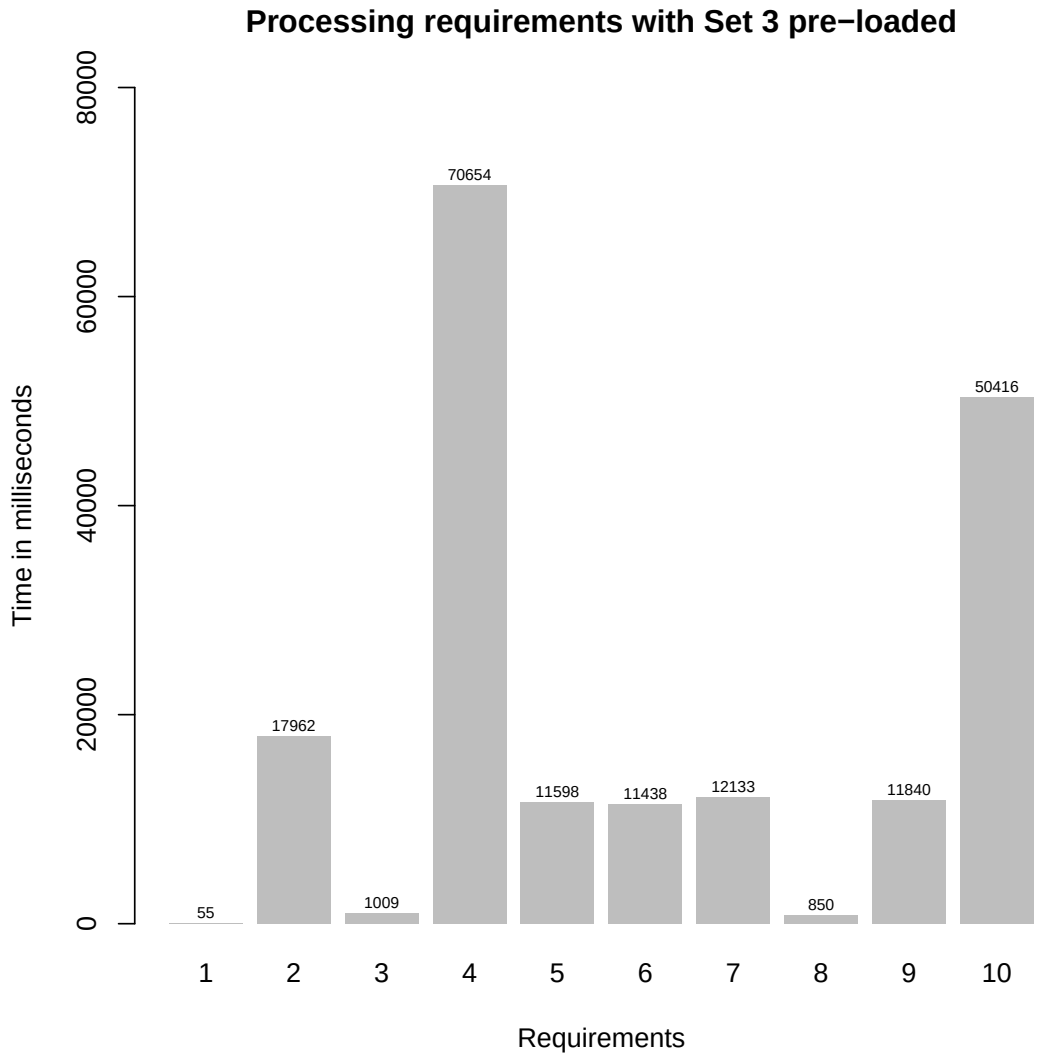


Figure 6.10 – Requirements sent against a framework configured with set numbered “3”.

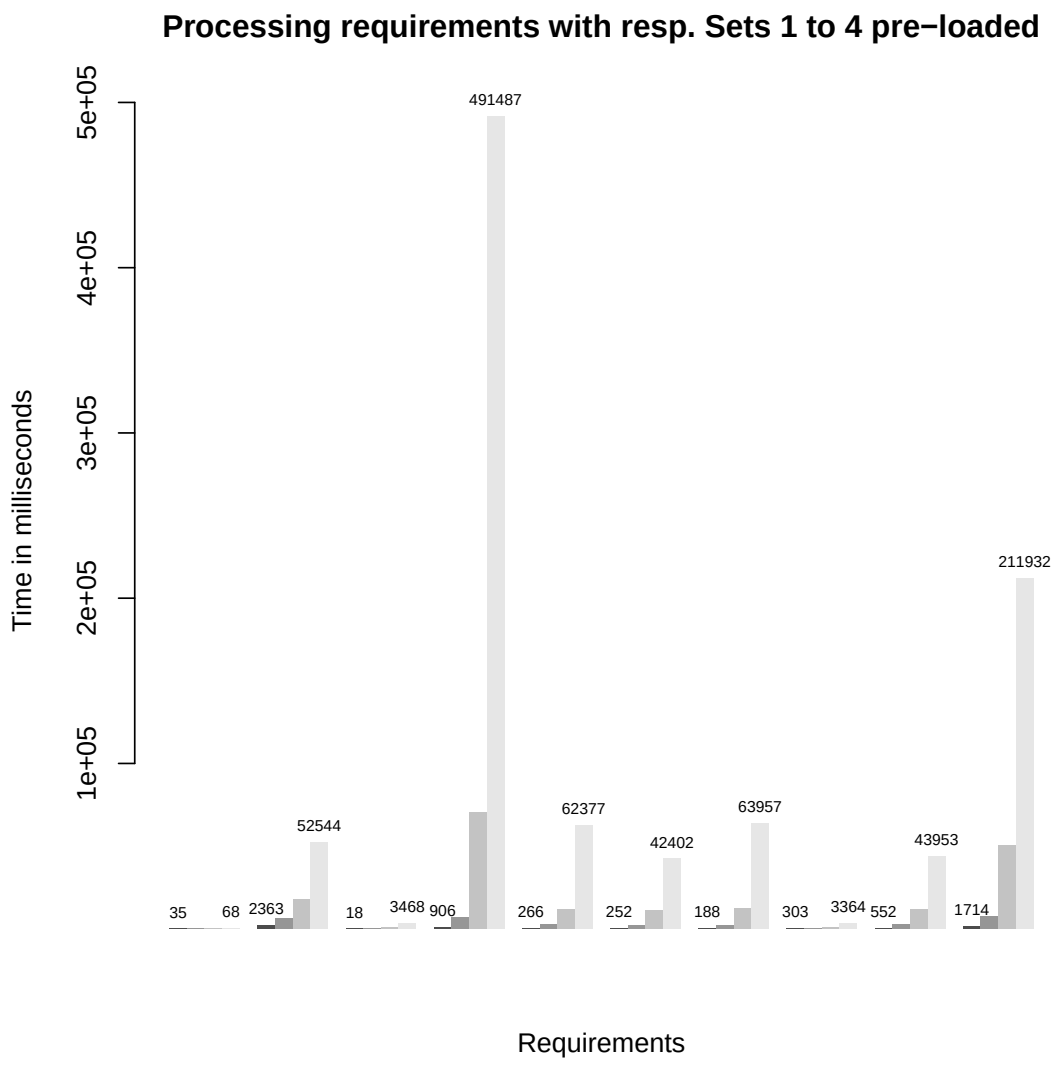


Figure 6.11 – Processing requirements in different configurations.

6.2.5 Assessing sharing message protocol of the federation

To evaluate a federated network of semantic nodes, the indoor location model (presented in Section 3.3) has been instantiated with different types of places, namely, floors, corridors and various types of rooms (offices, meeting rooms and labs), all in the same building. A node has then been deployed in each described premises to build up a federated architecture, comprising four levels of management (i.e. the maximum distance between the root and any leaf node). Our evaluation approach consists of testing the applicability of the implemented sharing knowledge mechanism through a scenario showing the feasibility of the approach by quantitatively evaluating the scalability of the federation.

Scenario Validation

The sharing knowledge mechanism has been applied to a scenario that is representative of dynamic IoT systems. The testbed consists of a number of sensors deployed in rooms in a building, with four floors in the building. We organized the testbed into a federated network of nodes, comprising up to 4 management levels (i.e. building, floor, open space and room). The distribution on a given floor is as shown in Figure 6.12 (gray circles represent a VR gateway acting as a semantic node and associated with a place). The deployment of the connected devices in each node triggers its *Processing and Storage* block which processes the corresponding semantic descriptions and stores them in the triple store. Once this is done for each node, the double cascading process (Section 5.2.2) allows the information related to the distribution of the nodes to be shared within the federation. The first case of the scenario consists of a nomadic user, Ben, who moves around the building and is interested in finding the relevant connected devices that can give him an idea of his ambient temperature at any given location. A user application allows to retrieve Ben's location (in terms of the Place Ben is currently roaming), contact the Gateway Resolver and setup Ben's VR Gateway client with the appropriate IP address of the VR Gateway to contact. Ben registers then to the VR Gateway, sending its user profile (referring to Chapter 3) further registered in a triple store associated to the semantic node. Finally, Ben sends a request to the *UserQuery* component of this VR Gateway, asking for sensor given information of type "Temperature" (this type typically refers to a sub-concept of Structure, as defined in Chapter 3, and could be defined as equivalent to the "ThermodynamicTemperature" concept defined in the QUDT [6] ontologies).

6. QUDT – Quantities, Units, Dimensions and Data Types Ontologies `http://www.qudt.org/`

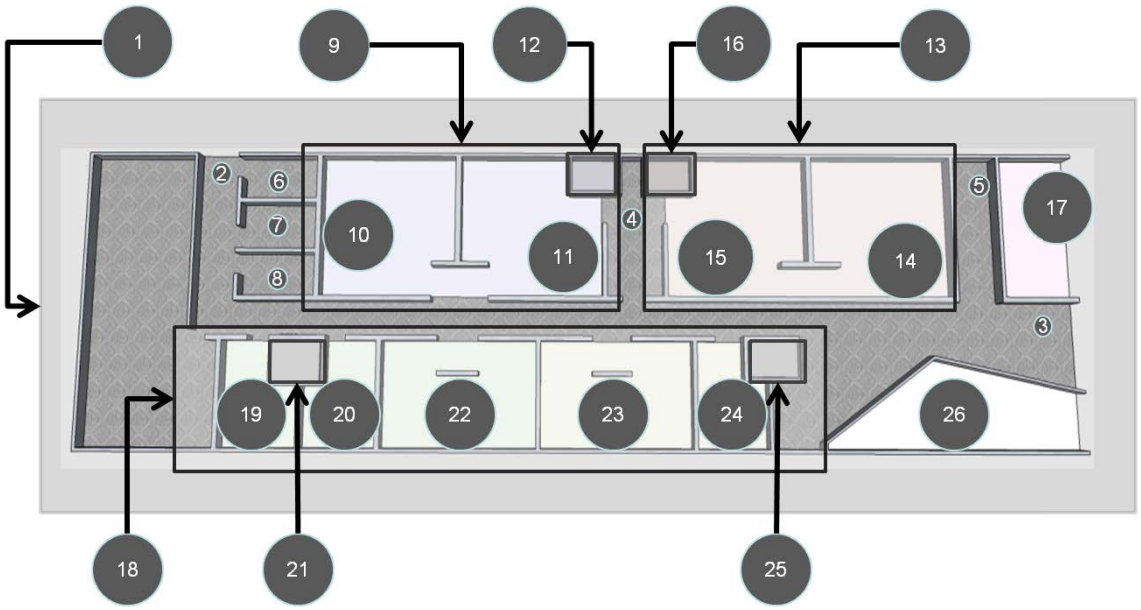


Figure 6.12 – Deployed semantic nodes in our building

This request then feeds the *Association Manager* component which tries finding matching sensors to the request of Ben, taking into account its profile and location. Since the room contains a temperature sensing service, it is associated to Ben. The result of this association is further stored in the Triple store of the node then shared with neighbor places to Ben localization.

The second case of the scenario showcases relocation of a connected device from one room to another, and thus a change in the Knowledge Base of several nodes. A generated event (connected device joining a place) triggers the *Association Manager* that tries finding new or updating existing associations. Next, the *Rule Manager* of the *Knowledge Propagation* block executes the relevant knowledge sharing rules to determine the set of nodes to be updated. The *Results Dispatcher* finally employs the notification algorithm to determine the path to the selected nodes and both the connected device representation and the newly determined (or updated) associations, are sent to these nodes.

Performance Measurements

Our evaluation approach consisted of a number of performance related experiments. The first experiment we performed was to assess the time taken to compute associations, by varying the number of connected devices to be taken into account by the *Association*

Manager, from 20 to 2000. We used a centralized triple store containing all the semantic descriptions of the considered connected devices. To determine associations, we also used a fixed set of four described physical quantities (all described in the QUDT ontology: Pressure, Temperature, Luminous Intensity and Length). Associations were then derived using the logic of the *Association Manager*. The results displayed in Figure 6.13 show the exponential growth of the time required to derive associations, in function of the number of connected devices. This experiment highlights the computationally expensive task of recomputing associations and validates the inappropriate use of a centralized approach to do so. As an example, Figure 6.13 shows that 20 seconds are required to recompute associations involving 200 connected devices, a number that may however be quickly reached when deploying devices in a whole building. This conclusion bolsters our belief that a federated architecture would be a more feasible deployment option in IoT-enabled Smart Environments, where each node would manage only a limited number of connected devices.

We assess the scalability of the federated framework by a second experimentation quantifying the number of messages exchanged with different nodes sharing information as well as the time taken to process these messages. For this experimentation, we used 20 nodes of the federated system associated to the Building displayed in Figure 6.12 and deployed 50 connected devices in each of them (i.e. the overall system was managing 1000 devices). We then simulated the relocation of groups of devices to evaluate how the number of relocations was impacting the federated system compared to a centralized approach. Tests involved respectively the relocation of 1, 20 and finally 50 devices. For this experimentation, we used a node sharing knowledge with only one other node. Consequently, respectively 1, 20 and 50 messages were generated. Upon receptions of these messages, semantic descriptions of relocated sensors were retrieved by the node and, finally, associations were derived. Figure 6.14 summarizes the overall times that we have obtained. These times are decomposed in the time taken to send the set of messages, the time taken to load the semantic descriptions associated to these messages and the time taken to recompute associations. This figure indicates that the time spent in sending messages follows a linear growth (function of the number of messages to send) resulting in a significant amount of time added by the knowledge sharing process. Besides, this figure shows that the time taken to load semantic profiles of connected devices was constant. Finally the time to compute associations follows a similar curve than what was presented in Figure

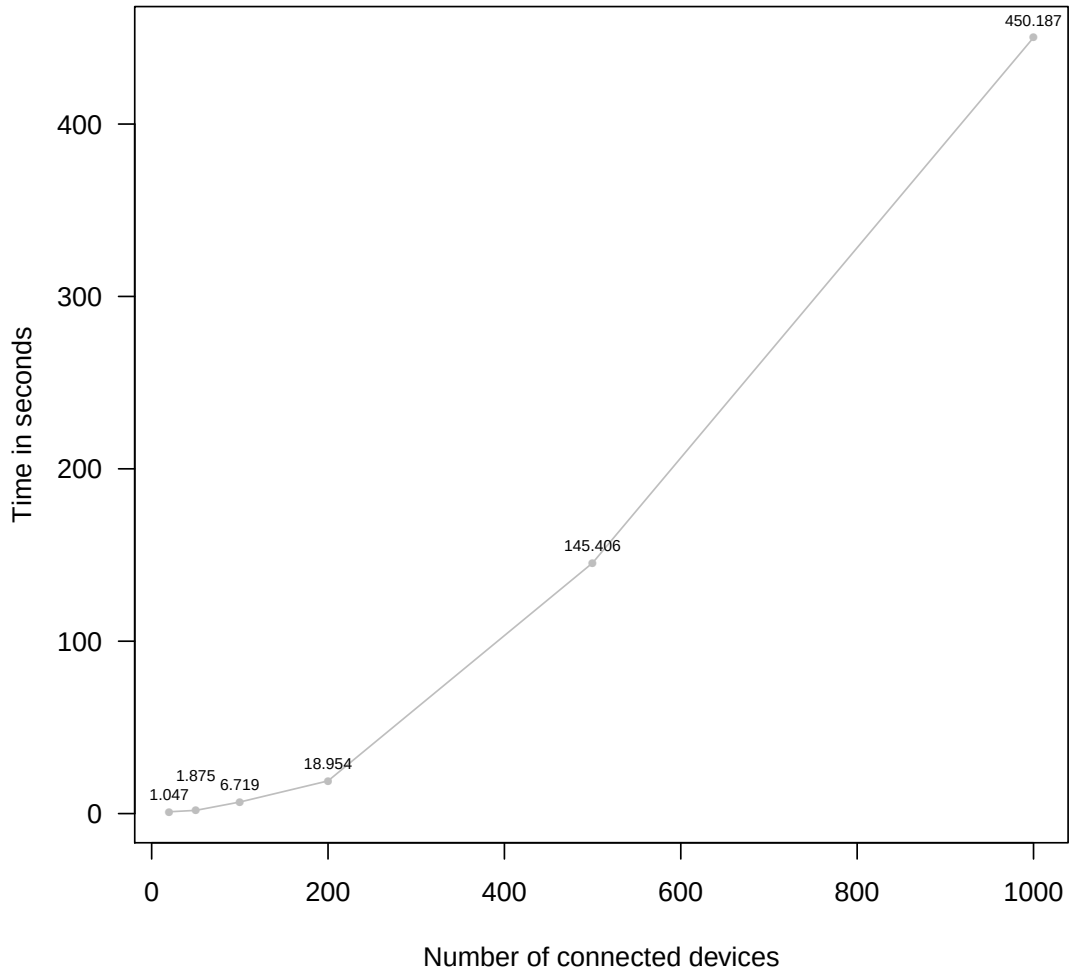


Figure 6.13 – Association computation measurements

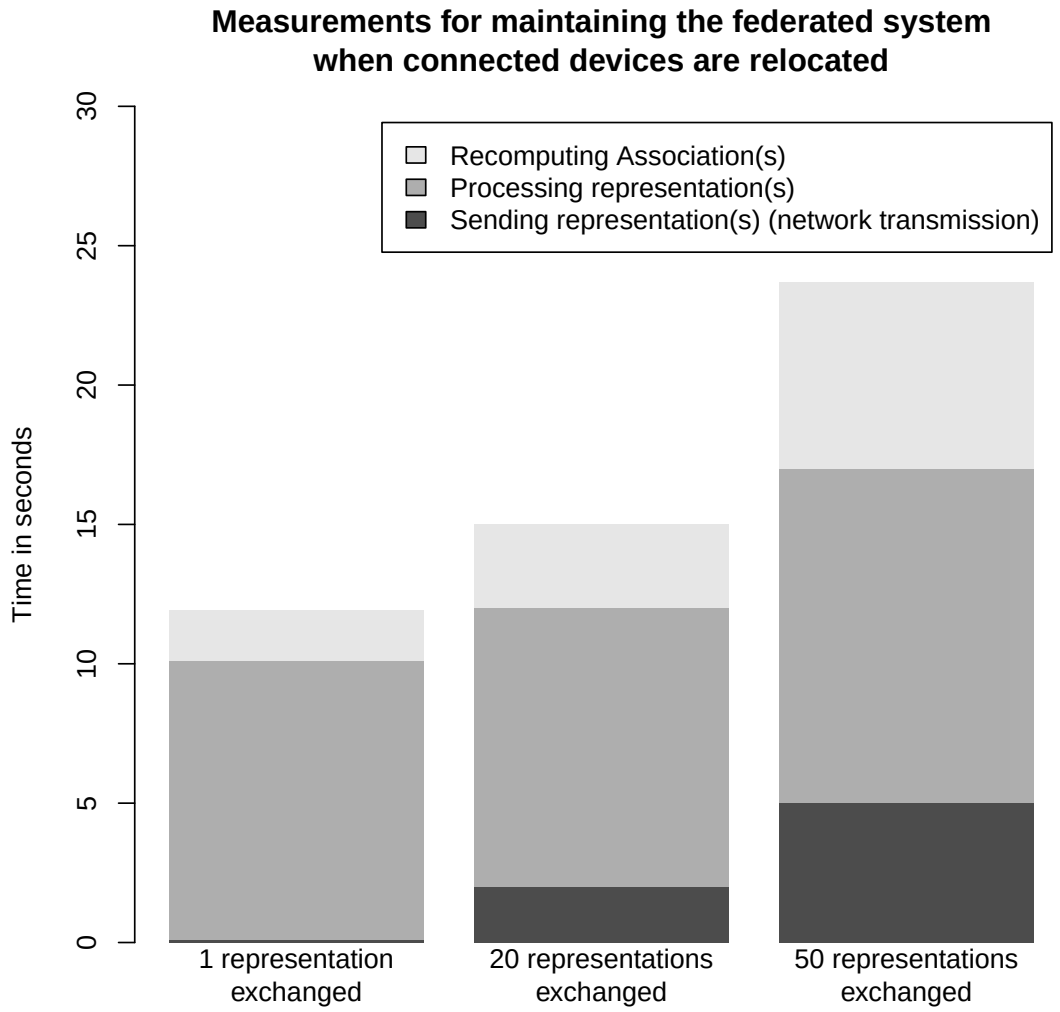


Figure 6.14 – Measurements for maintaining the federated system when connected devices are relocated

6.13. Compared to a centralized approach deriving associations with 1000 connected devices, these times stay however much more acceptable (see Figure 6.13 showing a time of 645 seconds to derive associations with 1000 connected devices).
Finally, we did a third experimentation checking whether the number of nodes crossed by a knowledge sharing message was impacting the federated system or not. We then run the scenario of the relocation of one connected device multiple times; varying the route of this relocation by changing the recipient room. Such scenario provided us with a set of messages, each having been propagated differently (i.e. having crossed up to 5 nodes). Although the time increased linearly with the number of nodes having been crossed, the results displayed in Figure 6.15 shows that it could be disregarded compared to others (i.e. time to load the semantic description of the relocated sensor and time to recompute associations using 50+1 representations of connected devices).

6.3 Conclusions

This chapter presents the instantiation and assessment of the ideas having been proposed in the three previous chapters. Embodied in a VR Gateway, these ideas refer to:

— the semantic similarity measure detailed in Chapter 4 and,
— the matching of applicative requirements against representations of connected devices exposed in Chapter 3.

Further deployed in a federated network, these VR Gateways implement the sharing protocol exposed in Chapter 5.
Based on this implementation, experimentations have been driven to assess the accuracy of the results returned by the semantic similarity measure, the time taken to answer to applicative requirements (when relying on DL reasoning) as well as the validity to create a distributed architecture instead of a centralized one in order to address scalability issues inherent to the growth in Internet of Things.
Some insights that can be derived from these experimentations validate different points tackled in this dissertation.
First, our semantic similarity measure seems to provide qualitatively better results than the state of the art when dealing with rich semantic descriptions and consequently may be a better approach when comparing representations of connected devices and applicative requirements, both of them likely to be underlied by the DL $\mathcal{SHOIQ}$ (see Chapter 3).

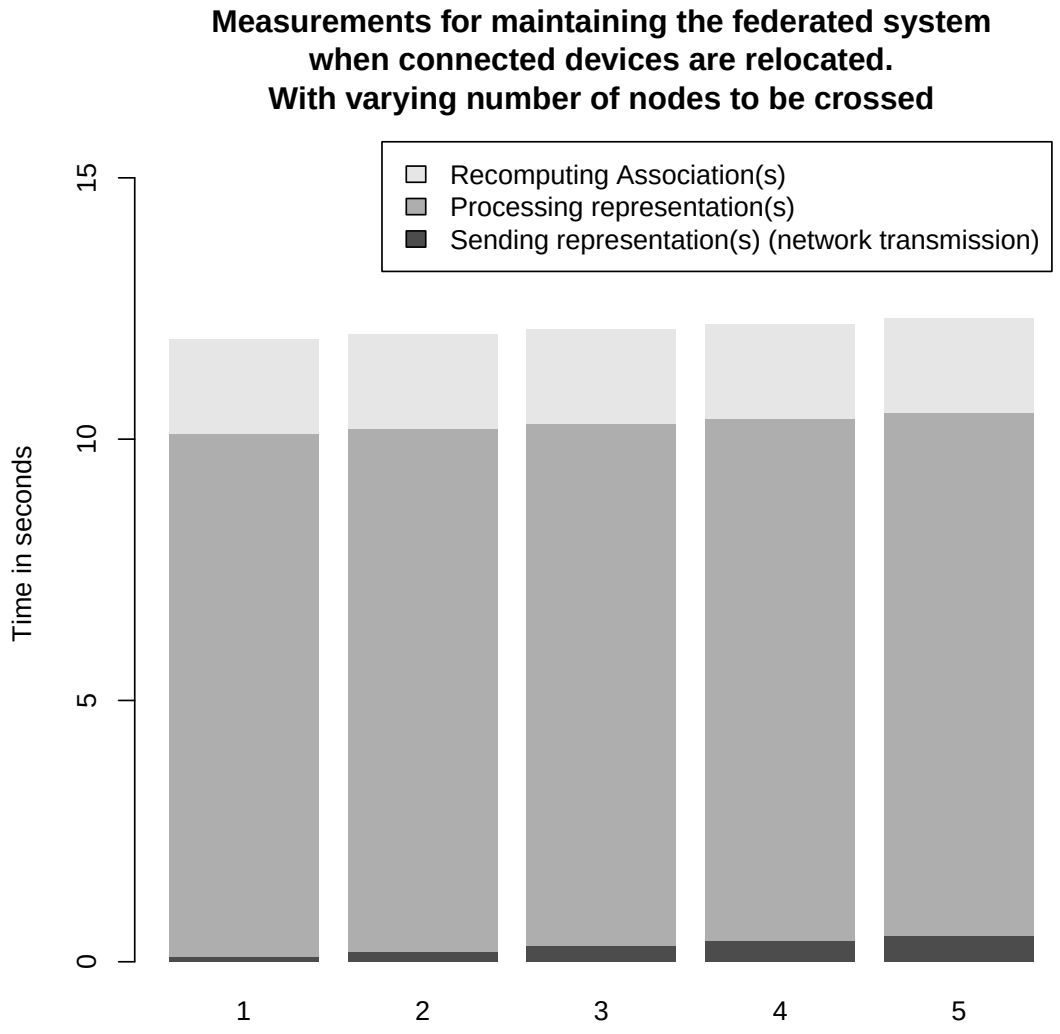


Figure 6.15 – Maintaining the federated system when one connected device is relocated

Aside the relevance of the results provided by the entailment of regime of a SPARQL-DL engine, the second experimentation highlights that the time taken to handle applicative requirements can drastically get longer by three factors: the absence of constraints in the query, the use of complex properties and expressions to form the query (transitive property, union, etc.) and the size of the data the engine has to deal with. Sounding like an impediment to the development of applications in IoT-enabled Smart Environments, this experimentation on the contrary allows writing recommendations to developers reducing the use of consuming queries as much as possible. Finally, this second experimentation emphasizes the strong limitations of the DL reasoning when dealing with a huge number of concepts defined with a rich logic. In particular, this second experimentation validates the idea of creating a distributed infrastructure of cooperating nodes empowered with search capabilities.

The last experiment focuses on the distributed infrastructure and shows that the time induced by the transmission of learnt results is ridiculous compared to the time spared by processing well delineated datasets.

Conclusions

Over the last few years, the realization of the early vision of Ubiquitous and Pervasive Computing coined by Weiser [Wei91] and Satyanarayanan [Sat01] has begun a tangible reality through the developments of smart environments, made possible by the growth in the Internet of Things and in particular, the rise of connected devices (sensors, actuators or other smart components).
Together with the preponderant place that smartphones take in the daily life of users, these nascent smart spaces pave the way to the development of novel types of interactions between the user and these smart spaces, typically embodied as applications carried by the phone of nomadic users and dynamically reconfiguring themselves by opportunistically making use of appropriate and available devices.

Summary of contributions

This dissertation contributes towards the realization of such applications by summarizing research efforts driven over the last six years and consisting of designing tools enabling the efficient selection of meaningful devices, i.e. *a service or data offered by one (or a composition of) device(s), satisfying user needs in terms of the functionality or information that it delivers. Thus, this service or data is accessible, available, aligned with the specificities of the user and, depending on the scenario is "localized" in the vicinity of the user.*
Research works presented in this dissertation revolve around the proposal of the eco-system resulting from the IoT as it is implemented today and from the concepts triggered by the Ubicomp community.
Based on the identification of the stakeholders of the eco-system as well as how they are interlinked, the dissertation highlights some of the challenges and proposes to dig into three research axes resulting from as many as studies having been driven:

Definition of Representation models. The diversity of connected devices, the services they offer, the data they generate, the peripherals that they use, their localization, their potential mobility, their associated ownership and access rights, their changing availability, etc. are as many as characteristics that call for representation models defining the associated semantics and computational details so that they will make possible the creation of pervasive applications. A formal approach based on the use of Description Logics is proposed to tackle this challenge. Overall three representation models are presented, one centered around the concepts defining a connected device, another one focusing on the expressions associated to applicative requirements and a last one going towards the definition of user profiles and preferences.

Design of a method to compute accurate semantic similarity measurements. The induced complexity associated with the aforementioned representation models drove our research efforts to the establishment of a process able to compute accurate semantic similarity measurements. Accordingly, we have defined a measure capable of an in-depth and refined analysis for any representation underlied by the Description Logic $\mathcal{SHOIQ}$. More specifically, the semantics conveyed in the representation models is elicited based on a family of generative functions that consist of building semantic neighborhoods composed of what we call "pseudo-concepts". Once retrieved, all the pseudo-concepts populate their original semantic model, allowing a classical Semantic Web reasoner to build a much more refined classification graph. A semantic similarity formula uses this expanded classification graph to cope with known limitations associated to the approaches of the state of the art. Indeed unlike intensional approaches, the method can be used even if the semantic model is devoid of instances or if it does not contain all possible instances of a given domain of interest. In other words, as soon as the concepts of a domain of interest have been defined in an ontology, our method can be used. Besides, the generative functions that we apply prior to the generation of the classification graph, enable our method to outperform extensional approaches that are not able to cope with ontologies underlied by a rich DL. Having solely relied on the theoretical foundations (Description Logics) underlying the representations of connected devices and applicative requirements gives another interest to our process. Indeed, whatever the domain of interest being addressed, the process can be used as is.

Design of a scalable search infrastructure. The high number of connected devices, as well as the heterogeneity of the data, services and peripherals that compose them, led some of our initial research works focusing on improving the interoperability between the aforementioned actors. Resulting to representation models making use of Semantic Web technologies, first experimentations highlighted however that complex applicative requirements were hard to process in a system having to handle a large collection of semantically described connected devices. This situation was further emphasized by the high mobility characterizing IoT systems where users, applications and devices are likely to roam across different smart environments. In particular, other experimentations shown the hardness to create a responsive system having to maintain (recompute) multiple associations (device-device, device-application, device-user). Consequently, we proposed to overcome this limitation by designing a distributed framework composed of nodes capable of processing Semantic Web descriptions and organized in a federated architecture. In addition to other approaches using Semantic Web technologies to design IoT systems, the approach taken in this dissertation considers a particular deployment infrastructure where each node of such infrastructure is mapped to a physical environment (e.g. a building, a room, etc.). Considering such a set of nodes allows reasoning mechanisms local to each node to be setup and enables knowledge to be locally stored, avoiding a single and centralized database (or Triple Store) to be flooded by queries (and data).

Ideas exposed in this dissertation have been embodied in a (set of) VR Gateway(s). Associated with a place (a building, a room, etc.), a VR gateway is capable of processing representations of connected devices, applications and users. Upon the reception of queries coming from users (resp. applicative requirements coming from applications) the VR Gateway selects the most appropriate connected devices, potentially taking into account user profile and preferences. The VR Gateway either uses semantic similarity computation or DL reasoning to return results. In case no results are found, the VR Gateway forwards the request to its nearby peers, so that the search space in which relevant devices can be found is expanded but still well-delineated to not return devices too far away from the localization of the place the request (applicative requirement) has been sent. Thus, the VR Gateway is based on RESTful principles and allows to interact between connected devices through a Web-based API. Experimentations passed on the VR Gateway have

allowed to assess the ideas summarized in this dissertation.

To summarize, this dissertation gathers six years of research efforts under the thematic of supporting nomadic users roaming across IoT-enabled Smart Environments; responding to the evolving trend of connecting devices on the Internet. Once made accessible to the masses such ideas, as well as the VR Gateway embodying them could lead to making a step ahead in the concrete use of Smart Environments and the realization of the vision envisioned by Mark Weiser.

Perspectives

Different kinds of perspectives can be envisioned whether one considers the contributions of this dissertation or the topic (the IoT) in which these contributions take place. The former would entail research works performed in the continuation of the studies presented in this manuscript as well as launch of public working groups to strengthen the adoption of the IoT. They could be seen as short or medium term works. Conversely, research perspectives relying on the IoT topic – and how it is implemented now – could be seen as longer term works. Tackling challenges that the IoT is currently facing (security, data processing, etc.), these works would require an in-depth investigation, first to understand the research barriers entailed by these challenges and second to find the technical ways to solve them.

Short and medium term perspectives

First of all, an expected output of this work accounts for fostering the adoption of the ideas exposed in this dissertation. A possible way toward this goal may lie in the creation of working groups into standard organizations (e.g., the W3C), to produce efficient models – refining the ones presented in this manuscript – adopted by a large community of different stakeholders (device manufacturers, application developers, etc.).
In terms of research work pursuing this thesis, one of them may consist of studying a particular mechanism to realize the filtering step of the searching mechanism exposed at the beginning of Chapter 4. A possible way in which research may be driven, could be the creation of a novel type of fuzzy inference system or – more probably – a novel way to feed an existing FIS.

Another improvement of the work presented in this dissertation may also rely on investigating different approaches to handle applicative requirements, i.e., relying on a richer semantic allowing the expression of complex statements.
Different strategies to distribute knowledge within a network of smart environments may also be studied (e.g., using a peer to peer scheme instead of a federation).
Last but not least, limitations brought by the use of Description Logics to represent profiles of connected devices and applicative requirements may also drive interesting research works. In particular the inability of Description Logics to capture the dynamics of a system, as well as the scalability issues that semantic engines often encounter when handling a huge number of descriptions, open the door for using and assessing different logics (e.g., allowing to represent temporalities) and algorithms.

Long term perspectives

With the advent of more and more connected devices, the IoT is no more "coming" but is already here. While the ramp-up has already begun, several reports point out that the adoption of the IoT intrinsically depends on the data generated by the plurality of connected devices (not only about mobile phones, tablets, laptops and wearables, but specialized sensors on people, clothing, cars, animals, houses, weather stations, video cams, drone flying machines, etc.). Indeed, the expected enormous number of connected devices, coupled with the sheer volume, velocity and structure of IoT data, creates challenges, particularly in the areas of security, privacy, data processing and network design. The following suggests some points in which research may be driven.

Security in the IoT. The increasing digitization and automation of the multitudes of devices deployed across different areas (smart environments, cities, etc.) are likely to create new security challenges. In particular, as it is expected that more data are going to transit between different points of the network, attacks focusing on obtaining unauthorized access to data being transmitted may happen. Putting in place the right security control mechanisms for IoT-based system becomes then a requirement. The challenge lies on the ability to define these requirements and to further bring technological components enabling to secure IoT systems. In particular, a technical barrier to overcome may rely on setting up a secure communication protocol between connected devices while most of them may be devoid of high computing capabilities.

Data protection in the IoT. As is already the case with smart-metering equipment and increasingly digitized automobiles, there will be a vast amount of data providing information on users' personal use of devices that, if not secured, can give rise to breaches of privacy. Applications in particular may be able to collect a wide range of data types about individuals roaming across different smart environments. The information collected may potentially give insights (or even reveal) user habits, location, interests and other preferences. Aggregated with other services and data, such information may even lead to infer knowledge on the user that would not have been found by examining data separately (similar to the Amalgamation paradox [7]). This is particularly challenging as the information generated by the IoT is a key to bringing better services. Research on anonymizing the data to avoid tracking users may therefore be a field of research to consider. The diversity of contexts in which IoT scenarios can take place suggests however the development of different approaches from what is found in the state of the art. Till now only few early research works (such that [MDG14]) seem going in this direction, indicating that this field of research is almost unexplored.

Data processing in the IoT. Processing all of the data in the IoT accounts for the capability to perform (at least) three steps. Data ingestion and cleansing (or the capability to harvest relevant but heterogeneous data coming from diverse devices and applications), data storage (potentially triggering the design of new types of databases) and real-time data analytics. If research works to allow data ingestion may be based on the ideas developed in this dissertation (semantic profiles), developing new design of databases as well as algorithms capable of extracting knowledge from data are two promising research areas that may require attention.

Elasticity of the network for IoT systems. Existing data center links are sized for the moderate-bandwidth requirements generated by human interactions with applications. The IoT promises to dramatically change these patterns by transferring massive amounts of small message sensor data to the data center for processing, dramatically increasing inbound data center bandwidth requirements.

7. Simpson's Paradox, `http://en.wikipedia.org/wiki/Simpson%27s_paradox`

Appendix A

Subsumption relations in SHOIQ

The following presents the different subsumption relations that ground the definition of the generative function family $\mathcal{F}_{\mathcal{G}}$. These properties involve the role hierarchy $\mathcal{R}$ of a KB as well as the logical operators of the DL $\mathcal{SHOIQ}$ (see Section 1.1.1).
For a given role $R \in \mathcal{R}$, we will note S^+ any of its super-property and S^- any of its subproperty. With these conventions, we have the following subsumption properties:
Subsumption properties on universal restrictions.

$$\begin{array}{lccccccc} P.1. & \top & \Rightarrow & \forall S^+.X & \sqsubseteq & \forall R.X & \sqsubseteq & \forall S^-.X \\ P.2. & X \sqsubseteq Y & \Rightarrow & \forall R.X & \sqsubseteq & \forall R.Y & & \end{array}$$

Proof. We will start by proving $\forall S^+.X \sqsubseteq \forall R.X$.
Suppose $\exists \alpha \in \forall S^+.X$ and $\alpha \notin \forall R.X$. Using De Morgan laws, we then have $\exists \alpha \in \forall S^+.X$ and $\alpha \in \exists R.\neg X$ which mean the following:

$$\alpha \in \forall S^+.X \Rightarrow S^+(\alpha, \beta) = \top \rightarrow \beta \in X, \tag{A.1}$$

$$\alpha \in \exists R.(\neg X) \Rightarrow \exists \beta \in \neg X / R(\alpha, \beta) = \top. \tag{A.2}$$

However, we also have:

$$R \sqsubseteq S^+ \Rightarrow \forall (\alpha, \beta), R(\alpha, \beta) = \top \rightarrow S^+(\alpha, \beta). \tag{A.3}$$

Hence, by Equation (A.2) and Equation (A.3), we have that:

$$\alpha \in \exists R.(\neg X) \Rightarrow \exists \beta \in \neg X / S^+(\alpha, \beta) = \top, \tag{A.4}$$

which contradicts Equation (A.1).

The same arguments can then be used to prove $\forall R.X \sqsubseteq \forall S^-.X$.

Finally, about P.2., suppose $\exists\alpha \in \forall R.X$ and $\alpha \notin \forall R.Y$. Again, by using De Morgan laws we have $\exists\alpha \in \forall R.X$ and $\alpha \in \exists R.\neg Y$ which means the following:

$$\alpha \in \forall R.X \Rightarrow R(\alpha, \beta) = \top \rightarrow \beta \in X, \tag{A.5}$$

$$\alpha \in \exists R.(\neg Y) \Rightarrow \exists\beta \in \neg Y / R(\alpha, \beta) = \top, \tag{A.6}$$

which obviously contradicts $X \sqsubseteq Y$ (as such β should exist in $X \sqcap \neg Y$ which is empty). □

Subsumption properties on existential restrictions.

$$\begin{array}{lccccccc} P.3. & \top & \Rightarrow & \exists S^-.X & \sqsubseteq & \exists R.X & \sqsubseteq & \exists S^+.X \\ P.4. & X \sqsubseteq Y & \Rightarrow & \exists R.X & \sqsubseteq & \exists R.Y & & \end{array}$$

Proof. By negating the expression of P.3. and applying De Morgan laws, we have:

$$\forall S^+.\neg X \sqsubseteq \forall R.\neg X \sqsubseteq \forall S^-.\neg X, \tag{A.7}$$

which is equivalent to P.1. that has been proved.

The same can be done for P.4., noticing that $X \sqsubseteq Y \Longleftrightarrow \neg Y \sqsubseteq \neg X$. We then have:

$$\neg Y \sqsubseteq \neg X \Rightarrow \forall R.\neg Y \sqsubseteq \forall R.\neg X, \tag{A.8}$$

which is equivalent to P.2. that has been proved. □

Subsumption properties on minimal cardinality restrictions.

$$\begin{array}{lccccccc} P.5. & \top & \Rightarrow & \geq nS^-.X & \sqsubseteq & \geq nR.X & \sqsubseteq & \geq nS^+.X \\ P.6. & X \sqsubseteq Y & \Rightarrow & \geq nR.X & \sqsubseteq & \geq nR.Y & & \end{array}$$

Proof. Again, we prove P.5. and P.6. by showing that the contrary is false.

For the left-side of P.5., suppose we do not have $\geq nS^-.X \sqsubseteq \geq nR.X$. Then it means that $\exists\alpha \in \geq nS^-.X$ and $\alpha \notin \geq nR.X$. Using De Morgan laws, we then have $\exists\alpha \in \geq nS^-.X$ and $\alpha \in < nR.X$. We then have the following:

$$\#\{x \in X / S^-(\alpha, x) = \top\} \geq n, \tag{A.9}$$

$$\#\{x \in X / R(\alpha, x) = \top\} < n. \tag{A.10}$$

However, with $S^- \sqsubseteq R$, we have $S^-(\alpha, x) \Rightarrow R(\alpha, x)$ ($\forall\alpha$ and x) that coupled with Equation (A.9), lead to the following:

$$\#\{x \in X / R(\alpha, x) = \top\} \geq n, \tag{A.11}$$

which contradicts Equation (A.10). As a consequence, the left-side of P.5. is true. The same deductions can be applied to prove $\geq nR.X \sqsubseteq \geq nS^+.X$ (as $R \sqsubseteq S^+$).
To prove P.6., supposing the contradiction is true, it holds that: $X \sqsubseteq Y$ and $\exists\alpha \in \geq nR.X$ and $\alpha \notin \geq nR.Y$. This leads to the following equations:

$$\#\{z \in X / R(\alpha, z) = \top\} \geq n, \tag{A.12}$$

$$\#\{z \in Y / R(\alpha, z) = \top\} < n. \tag{A.13}$$

Equation (A.12) accounts for saying that it exists at least n instances of X that are in relation with α through the property R. However, knowing that $X \sqsubseteq Y$, we know that all these instances are Y. As a consequence, Equation (A.12) and $X \sqsubseteq Y$ implies the following equation:

$$\#\{z \in Y / R(\alpha, z) = \top\} \geq n, \tag{A.14}$$

which contradicts Equation (A.13). As a consequence, P.6. is true. □

Subsumption properties on maximal cardinality restrictions.

$$\begin{array}{lrcccccc} P.7. & \top & \Rightarrow & \leq nS^+.X & \sqsubseteq & \leq nR.X & \sqsubseteq & \leq nS^-.X \\ P.8. & X \sqsubseteq Y & \Rightarrow & \leq nR.Y & \sqsubseteq & \leq nR.X & & \end{array}$$

Proof. By negating the expression of P.7. and applying De Morgan laws, we have:

$$> nS^-.X \sqsubseteq > nR.X \sqsubseteq > nS^+.X \tag{A.15}$$

The same reasoning than the one applied on P.5. can then be performed to prove P.7.
The same can be done for P.8., noticing that $X \sqsubseteq Y \iff \neg Y \sqsubseteq \neg X$. We then have:

$$\neg Y \sqsubseteq \neg X \Rightarrow > nR.Y \sqsubseteq > nR.X \tag{A.16}$$

Again the same reasoning that the one applied on P.6 can be performed to prove P.8. □

Subsumption properties on existential restrictions involving a transitive property and nominals.

$$
\begin{aligned}
P.9.\quad & R \in N_{R^+} \wedge \\
& \exists R.\{a_0, \cdots, a_n\} \wedge \\
& \forall \epsilon \in \{a_0, \cdots, a_n\}, \exists \eta \in \{b_0, \cdots, b_p\} \text{ such that } R(\epsilon, \eta) = \top \\
& \Rightarrow \\
& \exists R.\{a_0, \cdots, a_n\} \sqsubseteq \exists R.\{b_0, \cdots, b_p\}
\end{aligned}
$$

Proof. Again, using De Morgan laws we will prove that the opposite of the proposition is absurd.

Suppose we have $\alpha \in R.\{a_0, \cdots, a_n\}$ and $\alpha \notin R.\{b_0, \cdots, b_p\}$. We then have the following:

$$\exists \epsilon \in \{a_0, \cdots a_n\}/R(\alpha, \epsilon) = \top, \tag{A.17}$$

$$\forall \eta \in \{b_0, \cdots b_p\}/R(\alpha, \eta) = \bot. \tag{A.18}$$

However Equation (A.17) and the two conditions:

— $R \in N_{R^+}$ and,

— $\forall \epsilon \in \{a_0, \cdots, a_n\}, \exists \eta \in \{b_0, \cdots, b_p\}$ such that $R(\epsilon, \eta) = \top$

imply that $\exists \eta \in \{b_0, \cdots, b_p\}$ such that $R(\alpha, \eta) = \top$ which contradicts Equation (A.18). As a consequence, P.9. is true. □

Appendix B

Algorithms generating pseudo-concepts

Algorithm 2 Generate the pseudo-concepts $\mathcal{PS}$ of an ontology $\mathcal{O}$

$\mathcal{PS} \leftarrow \emptyset$
procedure GENERATE_PS($\mathcal{O}$):
    for all concepts $C \in \mathcal{O}$ **do**
        $M \leftarrow \emptyset$
        $C_{\mathcal{SHOIQ}} \leftarrow$ normalize(C)
    end for
    if $C_{\mathcal{SHOIQ}} \equiv \sqcup_i D_i$ **then**
        $\mathcal{PS} \leftarrow$ SUBSUMEES($C_{\mathcal{SHOIQ}}, M$)
    else
        $\mathcal{PS} \leftarrow$ SUBSUMERS($C_{\mathcal{SHOIQ}}, M$)
    end if
end procedure

Algorithm 3 Generate the pseudo-concepts subsumed by a concept C in $\mathcal{O}$

procedure SUBSUMEES(C, M):
  $\mathcal{PS} \leftarrow \underline{\Phi_1}(C, M) \bigcup \underline{\Phi_2}(C, M)$
end procedure

procedure $\underline{\Phi_1}(C, M)$
  for all $D \in \mathcal{G}_\mathcal{O}$ and $D \sqsubset C$ **do**
    $\mathcal{PS} \leftarrow D$
    for all $X \in$ SUBSUMEES(D, M) **do**
      $\mathcal{PS} \leftarrow X$
    end for
  end for
end procedure

procedure $\underline{\Phi_2}(C, M)$
  if $(C \equiv \forall R.D)$ or $(C \equiv \exists R.D)$ or $(C \equiv \geq nR.D)$ or $(C \equiv \leq nR.D)$ **then**
    $S^+ \leftarrow \{S \in N_R \ / \ R \sqsubseteq S\}$
    $S^- \leftarrow \{S \in N_R \ / \ S \sqsubseteq R\}$
  end if
  if $(C \equiv \forall R.D$ and $C \notin M)$ **then**
    for all $S \in S^+$ **do**
      if $(R \sqsubset S)$ **then**
        $\mathcal{PS} \leftarrow \forall S.D$
      end if
      for all $X \in$ SUBSUMEES$(D, M \sqcup \{C\})$ **do**
        $\mathcal{PS} \leftarrow \forall S.X$
      end for
    end for
  else if $(C \equiv \exists R.D)$ and $C \notin M$ **then**
    for all $S \in S^-$ **do**
      if $(S \sqsubset R)$ **then**
        $\mathcal{PS} \leftarrow \exists S.D$
      end if
      for all $X \in$ SUBSUMEES$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \exists S.X$
      end for
    end for
  else if $(C \equiv \geq nR.D)$ **then**
    for all $S \in S^-$ **do**
      if $(S \sqsubset R)$ **then**
        $\mathcal{PS} \leftarrow \geq nS.D$
      end if
      for all $X \in$ SUBSUMEES$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \geq nS.X$
      end for
    end for
  else if $(C \equiv \leq nR.D)$ **then**
    for all $S \in S^+$ **do**
      if $(R \sqsubset S)$ **then**
        $\mathcal{PS} \leftarrow \leq nS.D\}$
      end if
      for all $X \in$ SUBSUMERS$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \leq nS.X$
      end for
    end for
  else if $C \equiv \bigcup_{i=1}^{N} C_i$ and $C \notin M$ **then**
    for $k = 1 \cdots N$ **do**
      $\mathcal{PS} \leftarrow \bigcup_{l=1}^{k} C_{j_l}$ such that $1 \leq j_1 < j_l < j_k \leq N$
    end for
  end if
end procedure

Algorithm 4 Generate the pseudo-concepts that subsume a concept C in $\mathcal{O}$

procedure SUBSUMERS(C, M):
  $\mathcal{PS} \leftarrow \overline{\Phi_1}(C, M) \bigcup \overline{\Phi_2}(C, M)$
end procedure

procedure $\overline{\Phi_1}(C, M)$
  for all $D \in \mathcal{G}_\mathcal{O}$ and $C \sqsubset D$ **do**
    $\mathcal{PS} \leftarrow D$
    for all $X \in$ SUBSUMERS(D, M) **do**
      $\mathcal{PS} \leftarrow X$
    end for
  end for
end procedure

procedure $\overline{\Phi_2}(C, M)$
  if $(C \equiv \forall R.D)$ or $(C \equiv \exists R.D)$ or $(C \equiv \geq nR.D)$ or $(C \equiv \leq nR.D)$ **then**
    $S^+ \leftarrow \{S \in N_R \ / \ R \sqsubseteq S\}$
    $S^- \leftarrow \{S \in N_R \ / \ S \sqsubseteq R\}$
  end if
  if $(C \equiv \forall R.D$ and $C \notin M)$ **then**
    for all $S \in S^-$ **do**
      if $(S \sqsubset R)$ **then**
        $\mathcal{PS} \leftarrow \forall S.D$
      end if
      for all $X \in$ SUBSUMERS$(D, M \sqcup \{C\})$ **do**
        $\mathcal{PS} \leftarrow \forall S.X$
      end for
    end for
  else if $(C \equiv \exists R.D)$ and $C \notin M$ **then**
    for all $S \in S^+$ **do**
      if $(R \sqsubset S)$ **then**
        $\mathcal{PS} \leftarrow \exists S.D$
      end if
      for all $X \in$ SUBSUMERS$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \exists S.X$
      end for
    end for
    if $(R \in N_{R^+}$ and $D \equiv \{\epsilon_1 \cdots \epsilon_m\})$ **then**
      if $(X \equiv \{\eta_1 \cdots \eta_n\}$ and $\forall \epsilon_i \in D, \exists \eta_j \in X$ such that $\exists R.\{\epsilon_i\} \sqsubseteq \exists R.\{\eta_j\}$ and $\forall \eta_j \in X, \exists \epsilon_i \in D$ such that $\exists R.\{\epsilon_i\} \sqsubseteq \exists R.\{\eta_j\})$ **then**
        $\mathcal{PS} \leftarrow \exists R.X$
      end if
    end if
  else if $(C \equiv \geq nR.D)$ **then**
    for all $S \in S^+$ **do**
      if $(R \sqsubset S)$ **then**
        $\mathcal{PS} \leftarrow \geq nS.D$
      end if
      for all $X \in$ SUBSUMERS$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \geq nS.X$
      end for
    end for
  else if $(C \equiv \leq nR.D)$ **then**
    for all $S \in S^-$ **do**
      if $(S \sqsubset R)$ **then**
        $\mathcal{PS} \leftarrow \leq nS.D\}$
      end if
      for all $X \in$ SUBSUMEES$(D, M \sqcup C)$ **do**
        $\mathcal{PS} \leftarrow \leq nS.X$
      end for
    end for
  else if $C \equiv \bigcap_{i=1}^{N} C_i$ and $C \notin M$ **then**
    for $k = 1 \cdots N$ **do**
      $\mathcal{PS} \leftarrow \bigcap_{l=1}^{k} C_{j_l}$ such that $1 \leq j_1 < j_l < j_k \leq N$
    end for
  end if
end procedure

List of Figures

Bibliography

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, November 1983.

[All84] James F. Allen. Towards a general theory of action and time. *Artif. Intell.*, 23(2):123–154, July 1984.

[BBH+10] Claudio Bettini, Oliver Brdiczka, Karen Henricksen, Jadwiga Indulska, Daniela Nicklas, Anand Ranganathan, and Daniele Riboni. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.*, 6(2):161–180, April 2010.

[BBS04] Magdalena Balazinska, Hari Balakrishnan, and Mike Stonebraker. Contract-based load management in federated distributed systems. In *Proceedings of the 1st Conference on Symposium on Networked Systems Design and Implementation - Volume 1*, NSDI'04, pages 15–15, Berkeley, CA, USA, 2004. USENIX Association.

[BC08] Dario Bonino and Fulvio Corno. Dogont - ontology modeling for intelligent domotic environments. In Amit Sheth, Steffen Staab, Mike Dean, Massimo Paolucci, Diana Maynard, Timothy Finin, and Krishnaprasad Thirunarayan, editors, *The Semantic Web - ISWC 2008*, volume 5318 of *Lecture Notes in Computer Science*, pages 790–803. Springer Berlin Heidelberg, 2008.

[BCBT11] Mathieu Boussard, Benoit Christophe, Olivier Le Berre, and Vincent Toubiana. Providing user support in web-of-things enabled smart spaces. In Dominique Guinard, Vlad Trifa, and Erik Wilde, editors, *WoT*, page 11. ACM, 2011.

[BGM+10] Norbert Baumgartner, Wolfgang Gottesheim, Stefan Mitsch, Werner Retschitzegger, and Wieland Schwinger. Beaware! - situation awareness, the ontology-driven way. *Data Knowl. Eng.*, 69(11):1181–1193, 2010.

[BJH+04] R A Belecheanu, G Jawaheer, A Hoskins, J McCann, and T Payne. Semantic web meets autonomic ubicomp. In *The 3rd International Semantic Web Conference*, 2004. Event Dates: November 7 -11, 2004.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web. *Scientific American*, 284(5):34–43, May 2001.

[BP10] Sotiris Batsakis and Euripides G. M. Petrakis. Sowl: Spatio-temporal representation, reasoning and querying over the semantic web. In *Proceedings of the 6th International Conference on Semantic Systems*, I-SEMANTICS '10, pages 15:1–15:9, New York, NY, USA, 2010. ACM.

[BVAC13] A.J. Bermejo, J. Villadangos, J.J. Astrain, and A. Córdoba. Ontology based road traffic management. In Giancarlo Fortino, Costin Badica, Michele Mal-

geri, and Rainer Unland, editors, *Intelligent Distributed Computing VI*, volume 446 of *Studies in Computational Intelligence*, pages 103–108. Springer Berlin Heidelberg, 2013.

[CBH92] William W. Cohen, Alex Borgida, and Haym Hirsh. Computing least common subsumers in description logics. In *Proceedings of the tenth national conference on Artificial intelligence*, AAAI'92, pages 754–760. AAAI Press, 1992.

[CBL+11] Benoit Christophe, Mathieu Boussard, Monique Lu, Alain Pastor, and Vincent Toubiana. The web of things vision: Things as a service and interaction patterns. *Bell Labs Technical Journal*, 16(1):55–61, 2011.

[CBTB12] Benoit Christophe, Mathieu Boussard, Vincent Toubiana, and Olivier Le Berre. A semantics to define web templates for adaptive ubicomp applications. In *GreenCom*, pages 217–224, 2012.

[CFJ05] Harry Chen, Tim Finin, and Anupam Joshi. The soupa ontology for pervasive computing. In *Ontologies for Agents: Theory and Experiences*, pages 233–258. BirkHauser, 2005.

[CGZK04] Eleni Christopoulou, Christos Goumopoulos, Ioannis Zaharakis, and Achilles Kameas. An ontology-based conceptual model for composing context-aware applications. In *Proc. 6th. International Conference on Ubiquitous Computing*, 2004.

[Che03] Harry Chen. An intelligent broker for context-aware systems. In *In Adjunct Proceedings of Ubicomp*, pages 183–184, 2003.

[CHN+09] Michael Compton, Cory Henson, Holger Neuhaus, Laurent Lefort, Amit Sheth, Kerry Taylor, Arun Ayyagari, and David De Roure. A survey of the semantic specification of sensors. In *Proceedings of the 2nd International Workshop on Semantic Sensor Networks (SSN09) at ISWC 2009*, volume 522, pages 17–32. CEUR Workshop Proceedings, November 2009.

[Chr11] B. Christophe. Semantic profiles to model the "web of things". In *Semantics Knowledge and Grid (SKG), 2011 Seventh International Conference on Semantics, Knowledge and Grid*, pages 51 –58, oct. 2011.

[Chr12] Benoit Christophe. Managing massive data of the internet of things through cooperative semantic nodes. In *ICSC*, pages 93–100, 2012.

[CKY+11] Yun-Gyung Cheong, Yeo-Jin Kim, Seung Yeol Yoo, Hosub Lee, Sunjae Lee, Seung Chul Chae, and Hyun-Jin Choi. An ontology-based reasoning approach towards energy-aware smart homes. In *Consumer Communications and Networking Conference (CCNC), 2011 IEEE*, pages 850–854, Jan 2011.

[CNBC10] Michael Compton, Holger Neuhaus, Luis Bermudez, and Simon Cox. An ontology for sensor network. *Geophysical Research Abstracts*, 12(EGU2010-3817-1), 2010.

[CNS+06] Lorcan Coyle, Steve Neely, Graeme Stevenson, Mark Sullivan, Simon Dobson, Paddy Nixon, and Gaëtan Rey. Sensor fusion-based middleware for smart homes, 2006.

[CNW12] Liming Chen, C.D. Nugent, and Hui Wang. A knowledge-driven approach to activity recognition in smart homes. *Knowledge and Data Engineering, IEEE Transactions on*, 24(6):961–974, June 2012.

[CPLM11] Yongyun Cho, Sangjoon Park, Jongchan Lee, and Jongbae Moon. An owl-based context model for u-agricultural environments. In Beniamino Murgante, Osvaldo Gervasi, Andrés Iglesias, David Taniar, and BernadyO. Apduhan, editors, *Computational Science and Its Applications - ICCSA 2011*, volume 6785 of *Lecture Notes in Computer Science*, pages 452–461. Springer Berlin Heidelberg, 2011.

[d'A] Claudia d'Amato. Similarity-based learning methods for the semantic web.

[DD09] Danica Damljanovic and Vladan Devedzic. *Applying Semantic Web to E-Tourism*, pages 243–265. The Semantic Web for Knowledge and Data Management: Technologies and Practices. IGI Global, 8 2009.

[dFE05] C. d'Amato, N. Fanizzi, and F. Esposito. A semantic similarity measure for expressive description logics. In A. Pettorossi, editor, *Proceedings of Convegno Italiano di Logica Computazionale (CILC05) 21-22 June 2005, Rome, Italy*, 2005.

[dFE06] Claudia d'Amato, Nicola Fanizzi, and Floriana Esposito. A dissimilarity measure for ALC concept descriptions. In Hisham M. Haddad, editor, *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC 2006), April 23-27, 2006, Dijon, France*, pages 1695–1699. ACM, New York, NY, USA, 2006.

[Dij71] Edsger W. Dijkstra. A short introduction to the art of programming. August 1971.

[DLNN91] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Werner Nutt. The complexity of concept languages. In *Information and Computation*, pages 151–162. Morgan Kaufmann, 1991.

[DP14] Alessandra De Paola. An ontology-based autonomic system for ambient intelligence scenarios. In Salvatore Gaglio and Giuseppe Lo Re, editors, *Advances onto the Internet of Things*, volume 260 of *Advances in Intelligent Systems and Computing*, pages 1–17. Springer International Publishing, 2014.

[DSF08] Claudia D'Amato, Steffen Staab, and Nicola Fanizzi. On the influence of description logics ontologies on conceptual similarity. In *Proceedings of the 16th international conference on Knowledge Engineering: Practice and Patterns*, EKAW '08, pages 48–63, Berlin, Heidelberg, 2008. Springer-Verlag.

[ELES06] M. Eid, Ramiro Liscano, and A. El Saddik. A novel ontology for sensor networks data. In *Computational Intelligence for Measurement Systems and Applications, Proceedings of 2006 IEEE International Conference on*, pages 75–79, July 2006.

[FDE08] Nicola Fanizzi, Claudia D'Amato, and Floriana Esposito. Learning with kernels in description logics. In *Proceedings of the 18th international conference on Inductive Logic Programming*, ILP '08, pages 210–225, Berlin, Heidelberg, 2008. Springer-Verlag.

[FHHM01] Dieter Fensel, Frank Van Harmelen, Ian Horrocks, and Deborah L. Mcguinness. Oil: An ontology infrastructure for the semantic web. *IEEE Intelligent Systems*, pages 38–45, 2001.

[FK02] Josef Fink and Alfred Kobsa. User modeling for personalized city tours. *Artif. Intell. Rev.*, 18(1):33–74, September 2002.

[GANJ06] Yasser Ganjisaffar, Hassan Abolhassani, Mahmood Neshati, and Mohsen Jamali. A similarity measure for owl-s annotated web services. In *Web Intelligence*, pages 621–624. IEEE Computer Society, 2006.

[GD06] Dragan Gasevic and Vladan Devedzic. Petri net ontology. *Knowledge-Based Systems*, 19(4):220 – 234, 2006.

[GM03] Aldo Gangemi and Peter Mika. Understanding the semantic web through descriptions and situations. In *Proceedings of ODBASE03 Conference*, pages 689–706. Springer, 2003.

[GNOT92] David Goldberg, David Nichols, Brian M. Oki, and Douglas Terry. Using collaborative filtering to weave an information tapestry. *Commun. ACM*, 35(12):61–70, December 1992.

[Goo] Caleb Goodwin. An ontology-based sensor network prototype environment.

[GPZ04] Tao Gu, Hung Keng Pung, and Da Qing Zhang. A middleware for building context-aware mobile services. In *Vehicular Technology Conference, 2004. VTC 2004-Spring. 2004 IEEE 59th*, volume 5, pages 2656 – 2660 Vol.5, may 2004.

[Gru93] Thomas R. Gruber. A translation approach to portable ontology specifications. *Knowl. Acquis.*, 5(2):199–220, June 1993.

[GS04] Fabien L. Gandon and Norman M. Sadeh. Semantic web technologies to reconcile privacy and context awareness. *J. Web Sem.*, 1(3):241–260, 2004.

[Hay79] P. J. Hayes. The logic of frames. In D. Metzing, editor, *Frame Conceptions and Text Understanding*, pages 46–61. Walter de Gruyter and Co., Berlin, Germany, 1979.

[HB08] Matthew Horridge and Sean Bechhofer. The owl api: A java api for working with owl 2 ontologies. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *OWLED*, volume 529 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.

[HI04] Karen Henricksen and Jadwiga Indulska. Modelling and using imperfect context information. In *Pervasive Computing and Communications Workshops, 2004. Proceedings of the Second IEEE Annual Conference on*, pages 33–37. IEEE, 2004.

[HI06] Karen Henricksen and Jadwiga Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and mobile computing*, 2(1):37–64, 2006.

[HIR02] Karen Henricksen, Jadwiga Indulska, and Andry Rakotonirainy. Modeling context information in pervasive computing systems. In *Pervasive Computing*, pages 167–180. Springer Berlin Heidelberg, 2002.

[HKGB09] Jörg Henß, Joachim Kleb, Stephan Grimm, and Jürgen Bock. A Database Backend for OWL. In Rinke Hoekstra and Peter F. Patel-Schneider, editors, *Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009)*, volume 529, `http://ceur-ws.org`, 2009. CEUR Workshop Proceedings.

[HM85] Dennis Heimbigner and Dennis McLeod. A federated architecture for information management. *ACM Trans. Inf. Syst.*, 3(3):253–278, July 1985.

[HS05] Ian Horrocks and Ulrike Sattler. A tableaux decision procedure for shoiq. In *In Proc. of the 19th Int. Joint Conf. on Artificial Intelligence (IJCAI*, pages 448–453. Morgan, 2005.

[HWG07] Yuheng Hu, Zhendong Wu, and Ming Guo. Ontology driven adaptive data processing in wireless sensor networks. In *Proceedings of the 2Nd International Conference on Scalable Information Systems*, InfoScale '07, pages 46:1–46:2, ICST, Brussels, Belgium, Belgium, 2007. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

[HZHC12] Jing He, Yanchun Zhang, Guangyan Huang, and Jinli Cao. A smart web service based on the context of things. *ACM Trans. Internet Technol.*, 11(3):13:1–13:23, February 2012.

[ILMF11] Noha Ibrahim, Frédéric Le Mouël, and Stéphane Frénot. Semantic Service Substitution in Pervasive Environments. *International Journal of Services, Economics and Management (IJSEM)*, 2011. "Service-Oriented Engineering" special issue.

[IRRH03] Jadwiga Indulska, Ricky Robinson, Andry Rakotonirainy, and Karen Henricksen. Experiences in using cc/pp in context-aware systems. In *In Proc. of the Intl. Conf. on Mobile Data Management (MDM*, pages 247–261. Springer, 2003.

[Jac01] P. Jaccard. Distribution de la flore alpine dans le bassin des dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

[Jan93] J.-S.R. Jang. Anfis: adaptive-network-based fuzzy inference system. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(3):665–685, May 1993.

[Jan06] Krzysztof Janowicz. Sim-dl: Towards a semantic similarity measurement theory for the description logic alcnr in geographic information retrieval. In *SeBGIS 2006, OTM Workshops 2006. Volume 4278 of Lecture Notes in Computer Science*. Springer, 2006.

[JC97] J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.

[JMA10] Paul VanderLei Jeff McAffer and Simon Archer. *OSGi and Equinox: Creating Highly Modular Java Systems*. 2010.

[JT09] Xing Jiang and Ah-Hwee Tan. Learning and inferencing in user ontology for personalized semantic web search. *Inf. Sci.*, 179(16):2794–2808, July 2009.

[JW09] Krzysztof Janowicz and Marc Wilkes. Sim-dla: A novel semantic similarity measure for description logics reducing inter-concept to inter-instance similarity. In Lora Aroyo, Paolo Traverso, Fabio Ciravegna, Philipp Cimiano, Tom Heath, Eero Hyvönen, Riichiro Mizoguchi, Eyal Oren, Marta Sabou, and Elena Paslaru Bontas Simperl, editors, *ESWC*, volume 5554 of *Lecture Notes in Computer Science*, pages 353–367. Springer, 2009.

[KBM+02] Tim Kindberg, John J. Barton, Jeff Morgan, Gene Becker, Debbie Caswell, Philippe Debaty, Gita Gopal, Marcos Frid, Venky Krishnan, Howard Morris, John Schettino, Bill Serra, and Mirjana Spasojevic. People, places, things: Web presence for the real world. *MONET*, 7(5):365–376, 2002.

[KHF+11] Marc Kurz, Gerold Hölzl, Alois Ferscha, Alberto Calatroni, Daniel Roggen, Gerhard Tröster, Hesam Sagha, Ricardo Chavarriaga, José del R. Millán,

David Bannach, Kai Kunze, and Paul Lukowicz. The opportunity framework and data processing ecosystem for opportunistic activity and context recognition. *International Journal of Sensors, Wireless Communications and Control, Special Issue on Autonomic and Opportunistic Communications*, pages 102–125, December 2011.

[KKK+08] Artem Katasonov, Olena Kaykova, Oleksiy Khriyenko, Sergiy Nikitin, and Vagan Y. Terziyan. Smart semantic middleware for the internet of things. In Joaquim Filipe, Juan Andrade-Cetto, and Jean-Louis Ferrier, editors, *ICINCO-ICSO*, pages 169–178. INSTICC Press, 2008.

[Knu04] Holger Knublauch. Ontology-driven software development in the context of the semantic web: An example scenario with protege/owl. In David S. Frankel, Elisa F. Kendall, and Deborah L. McGuinness, editors, *1st International Workshop on the Model-Driven Semantic Web (MDSW2004)*, 2004.

[KS02] N.K. Kasabov and Qun Song. Denfis: dynamic evolving neural-fuzzy inference system and its application for time-series prediction. *Fuzzy Systems, IEEE Transactions on*, 10(2):144–154, Apr 2002.

[Las05] Ora Lassila. Applying semantic web in mobile and ubiquitous computing: Will policy-awareness help. In *in the proceedings of the Semantic Web and Policy Workshop, 4th International Semantic Web Conference*, 2005.

[LCM98] Claudia Leacock, Martin Chodorow, and George A. Miller. Using corpus statistics and wordnet relations for sense identification. *Computational Linguistics*, 24(1):147–165, 1998.

[LGSpt] J. Lathem, K. Gomadam, and A.P. Sheth. Sa-rest and (s)mashups : Adding semantics to restful services. In *Semantic Computing, 2007. ICSC 2007. International Conference on*, pages 469–476, Sept.

[Lin98] Dekang Lin. An Information-Theoretic Definition of Similarity. In Jude W. Shavlik and Jude W. Shavlik, editors, *ICML*, pages 296–304. Morgan Kaufmann, 1998.

[LLD] Fatiha Latfi, Bernard Lefebvre, and Céline Descheneaux. Ontology-based management of the telehealth smart home, dedicated to elderly in loss of cognitive autonomy.

[LZWQ05] Huiying Li, Xiang Zhang, Honghan Wu, and Yuzhong Qu. Design and application of rule based access control policies. In *Proc of the Semantic Web and Policy Workshop*, pages 34–41, 2005.

[Mam77] E. H. Mamdani. Application of fuzzy logic to approximate reasoning using linguistic synthesis. *IEEE Trans. Computers*, 26(12):1182–1191, 1977.

[mAYKGS09] Ching man Au Yeung, Lalana Kagal, Nicholas Gibbins, and Nigel Shadbolt. Providing access control to online photo albums based on tags and linked data. In *AAAI Spring Symposium: Social Semantic Web: Where Web 2.0 Meets Web 3.0*, pages 9–14. AAAI, 2009.

[MBH+04] David Martin, Mark Burstein, Erry Hobbs, Ora Lassila, Drew Mcdermott, Sheila Mcilraith, Srini Narayanan, Bijan Parsia, Terry Payne, Evren Sirin, Naveen Srinivasan, and Katia Sycara. Owl-s: Semantic markup for web services. Technical report, November 2004.

[MBKV+02] M. J. Martin-Bautista, D. H. Kraft, M. A. Vila, J. Chen, and J. Cruz. User profiles and fuzzy logic for web retrieval issues. *Soft Computing*, 6(5):365–372, 2002.

[MDEK13] Georgios Meditskos, Stamatia Dasiopoulou, Vasiliki Efstathiou, and Ioannis Kompatsiaris. Ontology patterns for complex activity modelling. In Leora Morgenstern, Petros Stefaneas, François Lévy, Adam Wyner, and Adrian Paschke, editors, *Theory, Practice, and Applications of Rules on the Web*, volume 8035 of *Lecture Notes in Computer Science*, pages 144–157. Springer Berlin Heidelberg, 2013.

[MDG14] Maria Laura Maag, Ludovic Denoyer, and Patrick Gallinari. Graph anonymization using machine learning. In *AINA*, pages 1111–1118, 2014.

[ME06] Brett McLaughlin and Justin Edelson. *Java and XML, 3rd Edition*. 2006.

[MFHS02] Deborah L. McGuinness, Richard Fikes, James Hendler, and Lynn Andrea Stein. Daml+oil: An ontology language for the semantic web. *IEEE Intelligent Systems*, 17(5):72–80, September 2002.

[MFSH03] Deborah L. McGuinness, Richard Fikes, Lynn Andrea Stein, and James A. Hendler. Daml-ont: An ontology language for the semantic web. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web*, pages 65–93. MIT Press, 2003.

[MH69] John Mccarthy and Patrick J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence*, pages 463–502. Edinburgh University Press, 1969.

[MPG+08] Sonia Ben Mokhtar, Davy Preuveneers, Nikolaos Georgantas, Valérie Issarny, and Yolande Berbers. Easy: Efficient semantic service discovery in pervasive computing environments with qos and context support. *J. Syst. Softw.*, 81(5):785–808, May 2008.

[MS02] Alexander Maedche and Steffen Staab. Measuring similarity between ontologies. In *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*, EKAW '02, pages 251–263, London, UK, UK, 2002. Springer-Verlag.

[NAA09] Hideyuki Nakashima, Hamid Aghajan, and Juan Carlos Augusto. *Handbook of Ambient Intelligence and Smart Environments*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[NK09] Abolghasem Sadeghi Niaraki and Kyehyun Kim. Ontology based personalized route planning system using a multi-criteria decision making approach. *Expert Syst. Appl.*, 36(2):2250–2259, March 2009.

[ORM+10] Benedikt Ostermaier, Kay Römer, Friedemann Mattern, Michael Fahrmair, and Wolfgang Kellerer. A real-time search engine for the web of things. In *Proceedings of Internet of Things 2010 International Conference (IoT 2010)*, Tokyo, Japan, November 2010.

[PB07] Michael J. Pazzani and Daniel Billsus. The adaptive web. chapter Content-based Recommendation Systems, pages 325–341. Springer-Verlag, Berlin, Heidelberg, 2007.

[PBW+04] Davy Preuveneers, Jan Van Den Bergh, Dennis Wagelaar, Andy Georges, Peter Rigole, Tim Clerckx, E Berbers, Karin Coninx, and Koen De Bosschere. Towards an extensible context ontology for ambient intelligence. In

In: Proceedings of the Second European Symposium on Ambient Intelligence, pages 148–159. Springer-Verlag, 2004.

[PE10] Giuseppe Pirrò and Jérôme Euzenat. A feature and information theoretic framework for semantic similarity and relatedness. In *International Semantic Web Conference (1)*, pages 615–630, 2010.

[PG11] F. Paganelli and D. Giuli. An ontology-based system for context-aware and configurable services to support home-based continuous care. *Information Technology in Biomedicine, IEEE Transactions on*, 15(2):324–333, March 2011.

[PL04] Terry R. Payne and Ora Lassila. Guest editors' introduction: Semantic web services. *IEEE Intelligent Systems*, 19(4):14–15, 2004.

[PRB+11] Dennis Pfisterer, Kay Römer, Daniel Bimschas, Oliver Kleine, Richard Mietz, Cuong Truong, Henning Hasemann, Alexander Kröller, Max Pagel, Manfred Hauswirth, Marcel Karnstedt, Myriam Leggieri, Alexandre Passant, and Ray Richardson. Spitfire: toward a semantic web of things. *IEEE Communications Magazine*, 49(11):40–48, 2011.

[RCC92] David A. Randell, Zhan Cui, and Anthony Cohn. A Spatial Logic Based on Regions and Connection. In Bernhard Nebel, Charles Rich, and William Swartout, editors, *KR'92. Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference*, pages 165–176. Morgan Kaufmann, San Mateo, California, 1992.

[Res95] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *IJCAI*, pages 448–453. Morgan Kaufmann, 1995.

[RMBB89] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *Systems, Man and Cybernetics, IEEE Transactions on*, 19(1):17–30, 1989.

[RMCM03] Anand Ranganathan, Robert E McGrath, Roy H. Campbell, and M. Dennis Mickunas. Use of ontologies in a pervasive computing environment. *Knowl. Eng. Rev.*, 18(3):209–220, September 2003.

[RNS+08] Michele Ruta, Tommaso Di Noia, Eugenio Di Sciascio, Floriano Scioscia, and Eufemia Tinelli. A ubiquitous knowledge-based system to enable rfid object discovery in smart environments. In *IWRT*, pages 87–100, 2008.

[Ros09] Sheldon Ross. *First Course in Probability, A (8th Edition)*. Prentice Hall, 8 edition, January 2009.

[Sat01] M. Satyanarayanan. Pervasive computing: Vision and challenges. *IEEE Personal Communications*, 8:10–17, 2001.

[Sch08] A. Schwering. Approaches to semantic similarity measurement for geospatial data: A survey. *Transactions in GIS*, 12(1):5–29, 2008.

[SCM10] Zhexuan Song, Alvaro A. Cárdenas, and Ryusuke Masuoka. Semantic middleware for the internet of things. In *2010 Internet of Things (IOT), IoT for a green Planet, Tokyo, Japan, November 29 - December 1, 2010. Proceedings*, 2010.

[Sid03] T. Sider. *Four-dimensionalism: An Ontology of Persistence and Time*. Mind Association Occasional Series. Clarendon Press, 2003.

[SKKR01] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of*

the 10th International Conference on World Wide Web, WWW '01, pages 285–295, New York, NY, USA, 2001. ACM.

[SLP04] Thomas Strang and Claudia Linnhoff-Popien. A context modeling survey. In *In: Workshop on Advanced Context Modelling, Reasoning and Management, UbiComp 2004 - The Sixth International Conference on Ubiquitous Computing, Nottingham/England*, 2004.

[SLPF03] Thomas Strang, Claudia Linnhoff-Popien, and Korbinian Frank. Cool: A context ontology language to enable contextual interoperability. In *LNCS 2893: Proceedings of 4th IFIP WG 6.1 International Conference on Distributed Applications and Interoperable Systems (DAIS2003). Volume 2893 of Lecture Notes in Computer Science (LNCS)., Paris/France*, pages 236–247. Springer Verlag, 2003.

[SM01] G. Stumme and A. Maedche. FCA–Merge: Bottom-Up Merging of Ontologies. In *IJCAI-2001 – Proceedings of the 17th International Joint Conference on Artificial Intelligence, Seattle, USA, August, 1-6, 2001*, pages 225–234, San Francisco, 2001. Morgen Kaufmann.

[SP07] Evren Sirin and Bijan Parsia. Sparql-dl: Sparql query for owl-dl. In *In 3rd OWL Experiences and Directions Workshop (OWLED-2007*, 2007.

[SPG+07] E. Sirin, B. Parsia, B. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical owl-dl reasoner. *Web Semantics: Science, Services and Agents on the World Wide Web*, 5(2):51–53, June 2007.

[SPL06] Sachin Singh, Sushil Puradkar, and Yugyung Lee. Ubiquitous computing: connecting pervasive computing through semantic web. *Inf. Syst. E-Business Management*, 4(4):421–439, 2006.

[STZ05] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, CIKM '05, pages 824–831, New York, NY, USA, 2005. ACM.

[Sus93] Michael Sussna. Word sense disambiguation for free-text indexing using a massive semantic network. In *Proceedings of the second international conference on Information and knowledge management*, CIKM '93, pages 67–74, New York, NY, USA, 1993. ACM.

[SVH04] Nuno Seco, Tony Veale, and Jer Hayes. An intrinsic information content metric for semantic similarity in wordnet, 2004.

[SVVB12] Thanos G. Stavropoulos, Dimitris Vrakas, Danai Vlachava, and Nick Bassiliades. Bonsai: A smart building ontology for ambient intelligence. In *Proceedings of the 2Nd International Conference on Web Intelligence, Mining and Semantics*, WIMS '12, pages 30:1–30:12, New York, NY, USA, 2012. ACM.

[SY93] M. Sugeno and T. Yasukawa. A fuzzy-logic-based approach to qualitative modeling. *Fuzzy Systems, IEEE Transactions on*, 1(1):7+, February 1993.

[TOD05] S. Ternier, D. Olmedilla, and E. Duval. Peer-to-peer versus federated search: Towards more interoperable learning object repositories. *Lecture Notes in Computer Science*, 2005.

[Tsu79] Y. Tsukamoto. *An Approach to Fuzzy Reasoning Method*. North-Holland Pub. Co., 1979.

[Tve77] A. Tversky. Features of similarity. *Psychological Review*, 84:327–352, 1977.

[Voo94] E.M. Voorhees. Query expansion using lexical-semantic relations. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '94), July 3-6, 1994, Dublin, Ireland*, pages 61–69. Springer New York Inc., NY, USA, 1994.

[Wei91] Mark Weiser. The computer for the 21st century. *Scientific American*, 265(3):66–75, September 1991.

[WF06] Chris Welty and Richard Fikes. A reusable ontology for fluents in owl. In *Proceedings of the 2006 Conference on Formal Ontology in Information Systems: Proceedings of the Fourth International Conference (FOIS 2006)*, pages 226–236, Amsterdam, The Netherlands, The Netherlands, 2006. IOS Press.

[WH11] Zachary Wemlinger and Lawrence Holder. The cose ontology: Bringing the semantic web to smart environments. In *Proceedings of the 9th International Conference on Toward Useful Services for Elderly and People with Disabilities: Smart Homes and Health Telematics*, ICOST'11, pages 205–209, Berlin, Heidelberg, 2011. Springer-Verlag.

[WP94] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *32nd. Annual Meeting of the Association for Computational Linguistics*, pages 133 –138, New Mexico State University, Las Cruces, New Mexico, 1994.

[WWG13] Anusha Indika Walisadeera, Gihan N. Wikramanayake, and Athula Ginige. An ontological approach to meet information needs of farmers in sri lanka. In *Proceedings of the 13th International Conference on Computational Science and Its Applications - Volume 1*, ICCSA'13, pages 228–240, Berlin, Heidelberg, 2013. Springer-Verlag.

[WZGP04] Xiao Hang Wang, Da Qing Zhang, Tao Gu, and Hung Keng Pung. Ontology based context modeling and reasoning using owl. pages 18–22, 2004.

[XMC13] Wenyi Xu, Christophe Marsala, and Benoit Christophe. Matching objects to user's queries in web of things' applications. In *Computational Intelligence for Communication Systems and Networks (CIComms), 2013 IEEE Symposium on*, pages 31–38. IEEE, 2013.

[Xu15] Wenyi Xu. *Modeling and exploiting the knowledge base of web of things*. PhD thesis, Université Pierre et Marie Curie-Paris VI, 2015.

[Zad75a] Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning - i. *Inf. Sci.*, 8(3):199–249, 1975.

[Zad75b] Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning - ii. *Inf. Sci.*, 8(4):301–357, 1975.

[Zad75c] Lotfi A. Zadeh. The concept of a linguistic variable and its application to approximate reasoning-iii. *Inf. Sci.*, 9(1):43–80, 1975.

[Zad84] L.A. Zadeh. Making computers think like people: The term 'fuzzy thinking' is pejorative when applied to humans, but fuzzy logic is an asset to machines in applications from expert systems to process control. *Spectrum, IEEE*, 21(8):26–32, Aug 1984.