NNT : 2024EMSEM015

# THÈSE DE DOCTORAT

## de Mines Saint-Etienne - Une école de l'IMT

**Ecole Doctorale N°488**
**(Sciences, Ingénierie, Santé)**

**Spécialité de doctorat :** Informatique

Soutenue publiquement le 8 avril 2024 par :

**Yousouf TAGHZOUTI**

# Négociation sémantique de contenu :

Une approche fine et souple utilisant les technologies du web sémantique

Devant un jury composé de :

| | | |
|---|---|---|
| Frédérique LAFOREST | Professeure, INSA Lyon | Présidente |
| Catherine FARON-ZUCKER | Professeure, Université Côte d'Azur | Rapporteuse |
| José Emilio LABRA GAYO | Professeur, Université d'Oviedo | Rapporteur |
| Hala SKAF-MOLLI | Professeure, Université de Nantes | Examinatrice |
| Antoine ZIMMERMANN | Professeur, Mines Saint-Etienne | Directeur de thèse |
| Maxime LEFRANÇOIS | Maître assistant, Mines Saint-Etienne | Encadrant |

# Affidavit

Je soussigné, Yousouf TAGHZOUTI, déclare par la présente que le travail présenté dans ce manuscrit est mon propre travail, réalisé sous la direction scientifique d'Antoine ZIMMERMANN, dans le respect des principes d'intégrité et de responsabilité inhérents à la mission de recherche. Les travaux de recherche et la rédaction de ce manuscrit ont été réalisés dans le respect de la charte nationale de déontologie des métiers de la recherche.

Ce travail n'a pas été précédemment soumis dans sa globalité en France ou à l'étranger dans une version identique ou similaire à un organisme examinateur.

Fait à Saint-Etienne, le 16 avril 2024

Cette thèse est rédigée en anglais. Cependant, nous fournissons un résumé en français des points clés en Annexe C.

This dissertation is written in English. However, we provide a French summary of the key points in Appendix C.

*To my parents, my siblings and my grandparents*

# Acknowledgment

In the first place, I would like to express my gratitude to my supervisors Antoine ZIMMERMANN and Maxime LEFRANÇOIS for their advice, encouragement and patience throughout my doctoral studies. I still remember my first interview with you from Poland during the COVID-19 pandemic. It was a very inspiring and stimulating experience to work with both of you.

Then, I would like to thank the members of the jury, Catherine FARON-ZUCKER, José Emilio LABRA GAYO, Frédérique LAFOREST and Hala SKAF-MOLLI for their valuable comments.

A special thank you to all the colleagues at Fayol who made this journey so unique. In particular, Olivier BOISSIER, Flavien BALBO and Marie-Line BARNEOUD for their administrative help and numerous teaching opportunities.

Moreover, I would like to thank colleagues from the Interactions Research Team at the University of St-Gallen who contributed to the success of my mobility there.

Equally, I would like to thank Antoine ISAAC who kindly participated in all of my thesis committees and helped guide my research.

Of course, a very special thank you goes to all of my friends from Algeria as well as those I met in Switzerland and France, especially Chikh Abd El Hakim and Idriss for their invaluable night classes.

All my love, and special thoughts to a few dear people who left us along the way. I can mention my uncles Khalou Hami, Amou Moulay and Fethi.

A distinct thank you to all my family members for their moral support and inestimable prayers. In particular, my siblings who have always been there for me.

Huge and profound appreciation goes to the most important duo of my life, my first mentors and advisors: my parents. Thank you for everything!

# Abstract

The World Wide Web started as an information management initiative with the aim of providing universal access to a vast set of interconnected resources. Resources in this space could have multiple representations that capture their state and vary along some dimensions. In such a case, web agents undertake a negotiation phase to select an appropriate representation. Besides, the Semantic Web adds structure and meaning to web content, making it a suitable area for collaboration between human and software agents. Concretely, resource representations are described using the Resource Description Framework by means of various ontologies and vocabularies. This has motivated the investigation of content negotiation in the profile dimension, which is orthogonal to the previous dimensions, where a profile is a description of semantic and/or structural constraints of a desired representation.

The aim of this thesis is to argue for the worth of a flexible (e.g. fine-grained constraints) and relaxed (e.g. in case of non-conformance of optional constraints) content negotiation taking into account that content negotiation is not a monolithic process, but rather composed of several stages: (1) discovery; (2) request formulation; (3) selection and/or adaptation; (4) response indication; (5) response interpretation.

First, we propose an abstract model for content negotiation we call CON-NEG (for CONtent NEGotiation). Then, we present the enhancements we made at each stage for the process: (1) discovering potential representations through equivalence links; (2) indicating finer constraints through severity; (3) relaxing validation; (4) reporting finer validation; (5) interpreting the validation report. After that, we introduce S-CON-NEG abstract model, for Semantic CON-NEG, which is its extension for flexible and relaxed content negotiation. Finally, we propose a typology of finer patterns for content negotiation in the studied profile dimension.

An instantiation of the abstract model is later proposed along with experiments to demonstrate the applicability and benefits of our methods. In addition, we illustrate how such improvements towards facilitating heterogeneous systems interaction by applying our approach in a Web-based multi-agent systems use case.

# Contents

# List of Figures

# List of Algorithms

# List of Tables

# Acronyms

**API** Application Programming Interface

**ARK** Archival Resource Key

**ASCII** American Standard Code for Information Interchange

**CBOR** Concise Binary Object Representation

**CC/PP** Composite Capability/Preference Profile

**CNTF** Content Negotiation Theoretical Framework

**CoAP** Constrained Application Protocol

**DNS** Domain Name System

**DXWG** Dataset Exchange Working Group

**FOAF** Friend Of A Friend

**FTP** File Transfer Protocol

**HTML** HyperText Markup Language

**HTTP** Hypertext Transfer Protocol

**IANA** Internet Assigned Numbers Authority

**IETF** Internet Engineering Task Force

**IoT** Internet of things

**IRI** Internationalized Resource Identifier

**JSON** JavaScript Object Notation

**MAS** Multi-Agent System

**MIME** Multipurpose Internet Mail Extension

**OGC** Open Geospatial Consortium

**OOP** Object-Oriented Programming

**OWL** Web Ontology Language

**PDF** Portable Document Format

**QSA** Query String Argument

**RDF** The Resource Description Framework

**RDFS** RDF Schema

**REST** Representational State Transfer

**RFC** Request for Comments

**RML** RDF Mapping language

**SHACL** Shapes Constraint Language

**ShEx** Shape Expressions

**SPIN** SPARQL Inferencing Notation

**UAProf** User Agent Profile

**URI** Uniform Resource Identifier

**URL** Uniform Resource Locator

**URN** Uniform Resource Name

**W3C** World Wide Web Consortium

**WAP** Wireless Application Protocol

**WOT** Web of Thing

**XML** Extensible Markup Language

Chapter 1

# Introduction

> I'm not arguing for anything except that the notion of *Semantic Content Negotiation* over HTTP might be worth exploring.
>
> *Danny Ayers 2006*

In 1989, Tim Berners-Lee made his proposal for a new information management system, "The World Wide Web", which we commonly call "the Web". It provides universal access to a vast set of interconnected resources [13]. A resource can be a tree, a person, or anything else. In addition, each resource can have different representations i.e. web documents, that capture its state. Take for example the tree species *Abies numidica*. It could have different representations that vary along several dimensions: media type such as HyperText Markup Language (HTML) or Portable Document Format (PDF); or language like English, French and Arabic. *content negotiation* is the mechanism used to select an appropriate representation from among those available. In 2001, Tim Berners-Lee, James Hendler and Ora Lassila published an article in the Scientific American magazine entitled "The Semantic Web" [21]. The article describes the Semantic Web as an extension of the Web that gives structure and meaning to the content available in this space. As a result, we have seen an evolution from a Web of documents to a Web of data.

Standardisation efforts by the Internet Engineering Task Force (IETF) and the World Wide Web Consortium (W3C) have contributed to making this vision a reality: Identifying, locating and accessing resources on the Internet is now well defined using Uniform Resource Identifier (URI) [18]. The Hypertext Transfer Protocol (HTTP) has been developed and refined for a distributed and collaborative hypermedia information system [58, 59, 61]. The core design components of the Web, as well as good practices for the principles and properties they support, are discussed in the specification of the Architecture of the World Wide Web [88]. The relationships between resources on the Web and their type are defined in Web Linking [129].

Similar standardisation efforts have been made in the area of the Semantic Web.

Initial proposals focused on The Resource Description Framework (RDF) [48, 82] and RDF Schema (RDFS) [32], which enable resource description. The Web Ontology Language (OWL) and its most recent version OWL 2 [73] have been proposed to allow the definition and modelling of more complex and richer knowledge. Furthermore, the need for a language to validate RDF data against a set of constraints motivated the development of Shapes Constraint Language (SHACL) [97] and Shape Expressions (ShEx) [137].

As a result, content negotiation may evolve; the client no longer negotiates only media type and language, but also structure and semantics. As early as 2006, discussions about vocabulary expectations and semantic content negotiation are found on the Semantic Web mailing list.[1] In 2017, the W3C chartered a working group (Dataset Exchange Working Group (DXWG)), one of its goals being to investigate content negotiation in a new dimension called *profile* [183]. In this context, the aim of this thesis is: (1) study content negotiation, and more specifically content negotiation by profile; (2) argue for flexible and relaxed content negotiation in this dimension, e.g. to define fine-grained constraints; and to handle the case of non-compliance to optional constraints. All this, taking into account that content negotiation is not a monolithic process, but consists of several stages: discovery, request formulation, selection and/or adaptation, response indication, response interpretation.

## 1.1 Motivation and Identified Limitations

The Web model, from its earliest days, provided the means to negotiate data format between content suppliers (servers) and requesters (clients), primarily to overcome interoperability problems, as described in the "WorldWideWeb - Executive Summary" [14]. This vision of content negotiation can be seen in the early versions of HTTP [15]. Negotiation over language, charset and encoding was introduced shortly afterwards [19, Section D.2]. In the course of the transition from the Web of documents to the Web of data, content negotiation has also evolved. However, there is a lack of an abstract model of the content negotiation problem that captures the negotiation process in the Web of documents when negotiating in dimensions such as media type and language. Such a model should also be extensible to current content negotiation practices in the Semantic Web. This is a serious limitation if we want to formalise the envisioned flexible and relaxed content negotiation.

**Limitation 1.** There is a need for a formalisation that reflects content negotiation in the Web of document, and the Web of data alike.

[1] https://lists.w3.org/Archives/Public/semantic-web/2006Jul/

When negotiating content in HTTP, quality values could be used to indicate the weight of different constraints [60, Section 3.9]. Such a technique is also used in later dimensions of content negotiation such as *profile* [161, 163]. However, in such a dimension the constraints could be finer. For example, a profile could be a SHACL shape graph and each shape could define multiple constraints. Yet, the weight is applied to the whole profile. Similarly, the server when handling the request, performing the validation, or responding, also could need finer techniques for reporting the content negotiation results, as existing status codes and HTTP response headers are not always sufficient. Besides, a client may have optional constraints and wish to convey such requirement. For example, a SHACL shape defining three constraints for three properties, the first being optional, the other two mandatory. Such semantics are neither specified nor handled in the content negotiation process. The all-or-none way of handling requests prevails.

**Limitation 2.** The formulation of the request and the handling of constraints are not sufficiently flexible and relaxed, nor are the validation reporting and the interpretation of the response.

Over the years, content negotiation has been used in various use cases ranging from device capabilities [36], archiving [91], to citation formatting [198]. Likewise, negotiating structure and semantics in the profile dimension is of practical need in multiple use cases [140]. However, even though different use cases may adopt the same content negotiation styles [61, Section 12], the actual implementation may differ in the use of finer patterns. Therefore, collecting and studying the various finer patterns is crucial for the profile dimension to gain acceptance and provide general and reusable solutions.

**Limitation 3.** There is no typography for finer patterns of content negotiation in the profile dimension.

Different representation providers use different identifiers for the same resource. Consequently, a client negotiating a resource representation with one of them is only considering a subset of all available representations. This means that if no representation satisfies the client's constraint on the current server, an alternative representation on another server may satisfy it.

**Limitation 4.** There is no means to negotiate distributed representations across multiple servers where appropriate.

The Web is not used by humans only, but also by software agents [25]. In addition, end users interact with servers through user agents such as browsers. Early involvement of such agents in the content negotiation process has been through sending pre-configured headers for a particular charset and encoding, or a language that the end user can understand. However, such software agents still lack the ability to locate negotiable resources and subsequently engage in content negotiation autonomously.

**Limitation 5.** Software agents are limited in their ability to engage in content negotiation autonomously.

Our thesis is that we can address the above limitations by, first, proposing an abstract model for content negotiation that reflects classical content negotiation behaviour in its original dimensions and, then, extending it to capture the requirements of a Semantic Web context. Second, using already available work in the Semantic Web space, such as equivalence links and semantic validation, enable more flexible and relaxed content negotiation that adheres to a well-defined finer patterns. And finally, equip software agents with the ability to engage in content negotiation autonomously where necessary.

## 1.2 Addressed Research Questions

The overarching objective of this dissertation is to contribute to the development of a semantic content negotiation framework in which web agents engage in more fine-grained and flexible content negotiation. We build our approach on state-of-the-art results from Semantic Web and multi-agent research. To achieve this framework, we address five research questions, which we present in the following paragraphs. Each question is focused on addressing one of the identified limitations.

A cornerstone of our thesis is the formalisation of the content negotiation problem. Content negotiation has been around for several decades and has been used in different scenarios and contexts, which raises the challenge of choosing the right level of abstraction, as well as the choice of elements to include in the abstract model in a way that makes it inclusive and extensible for future use (cf. Limitation 1). To this end, we introduce and formalise concepts that reflect current practice, and show how they fit into the HTTP protocol's way of specifying content negotiation. We then extend the general model in the context of the Semantic Web to allow the negotiation of representations conforming to some profiles.

**Research Question 1.** How can we formalise the problem of content negotiation in a general and extensible way to accommodate negotiation by profile?

Another key element in our thesis is the relaxation of content negotiation. For example, to be able to define fine-grained constraints; and to handle the case of non-compliance to optional constraints. However, this has to be done in several steps, since content negotiation consists of several stages (cf. Limitation 2). To this end, we provide solutions for the client to specify finer constraints through the use of constraint severities. Then, on the server side, we propose to relax the validation by considering these finer constraints to select adequate representations and report back the validation results to the client, which in turn can interpret and make use of such responses.

**Research Question 2.** How can a client and a server engage in a relaxed content negotiation process?

A key element in our thesis is the identification, collection, and the classification of finer patterns used when negotiating for content in the profile dimension (cf. Limitation 3). For this purpose, we introduce the notion of *content negotiation finer pattern*, which refers to patterns that answer more precise questions than content negotiation styles, such as, does the server use strict or flexible conformance checking? Is the conformance check performed on the fly or is it pre-processed? Is validation performed locally by the server or by a third party? We show how some of these content negotiation finer patterns can take advantage of current web standards and infrastructure.

**Research Question 3.** What are the finer patterns that can shape the negotiation process in the profile dimension?

On the Web, a resource could have different identifiers. Consequently, resource representations could be distributed over several servers, whereas regular content negotiation scenarios consider only one server (cf. Limitation 4). To this end, we propose to use equivalence links to discover potentially acceptable representations and adjust content negotiation architecture accordingly.

**Research Question 4.** How can we negotiate an appropriate variant of a resource when its representations are distributed?

Web clients, especially software agents, do not always have a priori knowledge of the content negotiation characteristics used in a web server, e.g. the content negotiation styles used in the negotiation, or the preferred means to convey constraints. Even when a documentation is available, it is usually designed for humans. An example of such documentation is a Swagger Application Programming Interface (API)

describing content negotiation expectations. As a result, it is difficult for software agents to engage in content negotiation with web servers (cf. Limitation 5). To this end, we propose the content negotiation ontology to capture knowledge about the content negotiation properties of web servers. We then illustrate how recent research from Multi-Agent System (MAS) can be used in accordance with the proposed ontology to enable autonomous agents to engage in semantic content negotiation.

**Research Question 5.** How can we enable autonomous agents to discover and exploit negotiable resources?

## 1.3 Dissertation Structure and Outline

This dissertation is structured in four parts:

In Part I, we analyse the state of the art to highlight the beginning and evolution of content negotiation on the Web and related work and technologies that can be used to bring about the envisioned semantic content negotiation framework.

In Chapter 2 we provide the background knowledge needed for the rest of the thesis. We begin by introducing the content negotiation problem in the light of the emergence of the Web. We then review the Semantic Web, and its components used throughout the thesis.

In Chapter 3 we first review the different content negotiation stages as well as how prior works have tried to formalise it. Then, we discuss the different content negotiation characteristics: dimensions, styles and constraint conveyance means. After that, we take a closer look at the profile dimension, concentrating on two main questions: how can a client specify a profile? and how can preferences be conveyed in this content negotiation dimension?

In Part II, we present our contributions for designing a flexible and relaxed content negotiation. In Chapter 4 we address Research Question 1. We present our proposed general model for content negotiation through a use case, starting with the case that reflects current practices, and discuss how it fits into the HTTP's way of specifying content negotiation. We then show how we have extended the general model in the context of the Semantic Web to allow content negotiation in the profile dimension.

In Chapter 5 we address Research Question 2. We describe how we enhance the content negotiation process by utilising semantic validation and being guided by the content negotiation stages: first by indicating finer constraints, then by relaxing the

validation and reporting finer validation, and finally by interpreting the validation report.

In Chapter 6 we address Research Question 3. We discuss possible finer patterns for handling content negotiation in the profile dimension. In particular, we show how some of them can make use of current web standards and infrastructure, and illustrate these using a use case and the current efforts of the DXWG and the IETF.

In Part III, we present the evaluation of our work. In Chapter 7 we address Research Question 4. First, we examine the representation distribution problem in more detail. Then, we present an approach to perform content negotiation when representations are distributed by leveraging equivalence links involving on-the-fly conformance checking. To this end, we provide two algorithms: The first in a basic context (i.e. considering only media type constraints) and the second in a semantic context (i.e. considering profiles). An implementation of the algorithms as well as separate experiments were conducted to measure the benefits and assess the time requirements of such methods.

In Chapter 8 we address Research Question 5. We start by giving an overview of concepts from MAS research that would be relevant to enabling semantic content negotiation in such systems. Then, we discuss our contribution towards facilitating the interaction of heterogeneous systems. First, we describe a Web-based Multi-Agent Systems use case where semantic content negotiation is needed. After that, we present the content negotiation ontology. Finally, we illustrate the steps to bring about and implement such an idea.

In Part IV, we provide a summary of our work and highlight directions for future research.

# Part I

State of the Art

Chapter 2

# Web of Document and Web of Data

## Contents



In this chapter, we present the main concepts and basic definitions necessary to understand the state of the art and the approaches proposed in the subsequent chapters.

First and foremost, in our dissertation we focus on the problem of *Content Negotiation* (sometimes called *Conneg* or *CN*) on the Web. More specifically, we address the problem of *Semantic Content Negotiation*, primarily within the HTTP protocol. Put simply: "two web agents, one serving semantic content and the other consuming it, negotiate the best available options given their respective semantic constraints".

However, prior to this, we need to provide an overview of the design of the Web and its architecture (in Section 2.1). Then we need to revisit the Semantic Web and

its main concepts, which will be used throughout the thesis (in Section 2.2). Finally, we examine how content negotiation is introduced and used in the HTTP protocol (in Section 2.3).

## 2.1 The Web and its Architecture

One of the important events of the 1990s in the field of information technology was the development and maturation of the Web project [42]. Its success is largely due to a well thought architecture and the properties it exhibits [88]. In Sections 2.1.1, 2.1.2 and 2.1.3 we give a brief overview of the core design components of the Web, namely identification, interaction and representation respectively. We then present the general architectural principles that apply to all of these architectural bases in Section 2.1.4.

It is worth noting that since the emergence of the Web there have been several architectural style propositions [62, Chapter 3], such as data flow and replication styles [5], as well as the widely used Representational State Transfer (REST) [62, Chapter 5]. However, in what follows, we have chosen to focus our discussion on the standard which introduces the architecture of the Web [88] because (1) it is an official standard specification and a W3C recommendation; (2) and to stay at a high level of abstraction and not be tied to specific architectural styles.

### 2.1.1 Identification

The Web is an information space where resources can be interconnected. A resource is anything worth referencing. However, in order to link resources together, one must first identify them, and such identification should be global, as the Web seeks to describe resources on a worldwide scale. For this reason, Web resources are identified by means of URIs [20]. Uniform Resource Locators (URLs) [22] and Uniform Resource Names (URNs) [149] are two subsets of URIs, meaning that every URL is a URI, but not every URI is a URL. The same thing with URNs. A URL specifies the location of a resource, while a URN specifies its name. The W3C/IETF URI planning interest group published a report offering clarifications and recommendations on their use [44]. Since URIs support only American Standard Code for Information Interchange (ASCII) encoding, Internationalized Resource Identifiers (IRIs) [52] were proposed to expand the set of permitted characters. Every IRI can be mapped to a URI for backwards-compatibility. Figure 2.1 shows an example of a URI with its component parts.

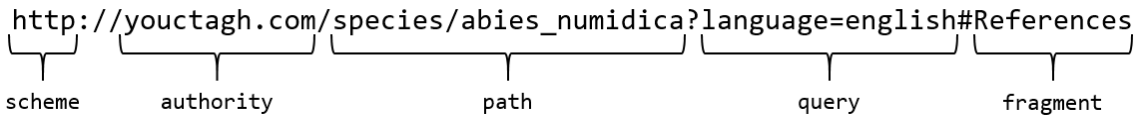



Figure 2.1: A URI example with its component parts.

The constraint of assigning different URIs to different resources must apply to avoid ambiguity and URI collisions. By analogy with our real-world identification of people, the same person could have different names (full name, first name only, nicknames, etc.). The Web architecture allows more than one URI to be associated with the same resource. Such URIs are called URI aliases. However, this situation raises the question of how to assert that two URIs identify the same resource or not. The URI specification proposes ways for normalising and comparing them [20, Section 6] such as through simple string comparison by considering the URIs under comparison as character strings.

### 2.1.2 Interaction

The Web is a distributed system that provides the means to retrieve information [17]. Consequently, web agents in this space need to communicate and interact with each other. Common web agents include: web clients (a.k.a. clients, the programs that establish connections to send requests, and expect responses), web servers (a.k.a. servers, the programs that accept incoming connections in order to service requests and send back responses), user-agent (e.g. browsers that retrieve and facilitate end-user interactions with web content), intermediary web programs (a.k.a. third parties, the programs involved in the interaction between user-agents and servers, such as proxies[1] and gateways[2]) [58, Section 1.3]. While all of these types of agents are present on the Web, for the sake of simplicity we will mainly focus on the architecture containing only clients and servers in our examples.

Interaction takes place according to Internet Application protocols (early examples of widely used protocols being the File Transfer Protocol (FTP) [136] and the Simple Mail Transfer Protocol [135]). Along with the proposal of the Web, a new protocol was introduced called HTTP for Hypertext Transfer Protocol [19].[3] HTTP is

[1] A proxy is a message-forwarding agent that a client chooses to pass through when requesting a resource from a server. One of its main uses is to increase privacy and security.

[2] A gateway (a.k.a reverse proxy) accepts clients' requests and forwards them to another server or servers. One of its main uses is to improve performance and scalability.

[3] HTTP version 1.0 [19] was first presented in a Request for Comments (RFC) in 1996. This

an application-level protocol for hypertext[4] information systems with the distinction of being stateless, meaning that each request/response interaction can be understood in isolation, without the need for previous ones.

With HTTP, clients can dereference a URI i.e. access the identified resource, to obtain a representation of the resource state. In a successful interaction, servers typically respond with a message containing a representation of the resource (more details about representation in Section 2.1.3).

### 2.1.3 Representation

Having the previous two components, web agents are able to identify a resource (cf. Section 2.1.1), and have the means to interact with one another using some protocol (cf. Section 2.1.2) to request some useful information about the resource state. A representation is the data that encodes information about the state of that resource, in combination with the representation metadata that describes that data. In summary, we have a URI that identifies a resource which is represented by a combination of data plus its metadata referred to as a representation. To illustrate these concepts, consider the following story.

**Story 1.** *YoucTagh* is a tree enthusiast and wants to create a website where he gives curated information about his favourite tree species. He chooses the URI template `http://youctagh.com/species/{species_name}` to identify them, and intends to propose the content in HTML. He starts with the first species, *Abies numidica*.

In this scenario, we have the resource *Abies numidica* identified by the URI `http://youctagh.com/species/abies_numidica`. It has an HTML representation that is served when dereferencing this URI. Figure 2.2 shows the relationship between these concepts. Listing 2.1 provides a snippet from the HTML representation of the *Abies numidica* tree.

specification was updated to version 1.1 in 1997 [60] and is still updated periodically [59, 61]. The HTTP protocol as it was first implemented is described at: `https://info.cern.ch/hypertext/WWW/Protocols/HTTP.html`

[4]Hypertext is a word coined by Theodor Holm Nelson that means "... a body of written or pictorial material interconnected in such a complex way that it could not conveniently be presented or represented on paper." [128]. On the Web, a hypertext document contains links that reference other hypertext documents, thus making them interconnected.

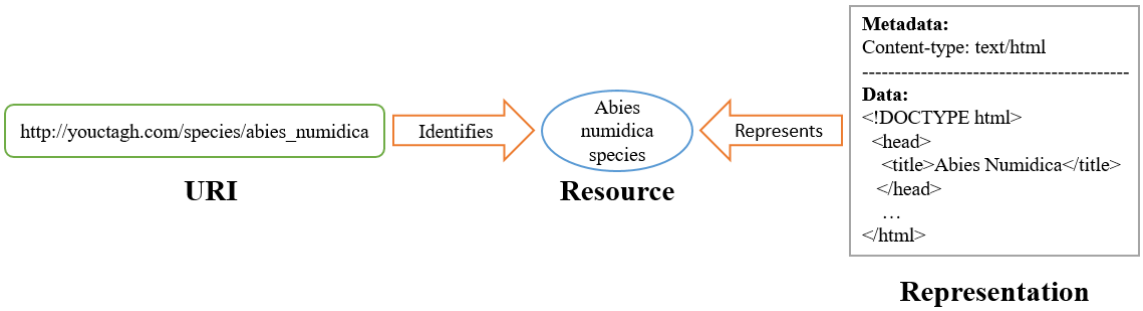



Figure 2.2: The relationship between the concepts of the Web architecture.

Listing 2.1: An extract from the HTML representation of the *Abies numidica* tree

```
1  <!DOCTYPE html>
2  <html lang="en">
3    <head> <title>Abies numidica</title> </head>
4    <body>
5      <a href="http://youctagh.com/species/abies_numidica">
6        Abies numidica
7      </a>
8      is a
9      <a href="http://youctagh.com/plant/tree">
10       tree
11     </a>
12     growing between 25 and 35 meters.
13   </body>
14 </html>
```

In this scenario we are using HTTP and we can see HTTP headers which are the means to express representation metadata, e.g. `content-type` to indicate the format.

*Media types* (formerly known as Multipurpose Internet Mail Extension (MIME) types) are identifiers used to specify the formats of the content being transmitted. In the given example, `text/html` is the media type identifying the HTML format. Internet Assigned Numbers Authority (IANA) is the official authority in charge of their registration and publication [63].

On the Web, agents are heterogeneous and, in particular, they do not have the same capabilities, and it is advisable for content providers to propose multiple representations that are more appropriate. This is achieved by separating content, presentation and interaction [88, Section 4.3]. In this way, multiple representations

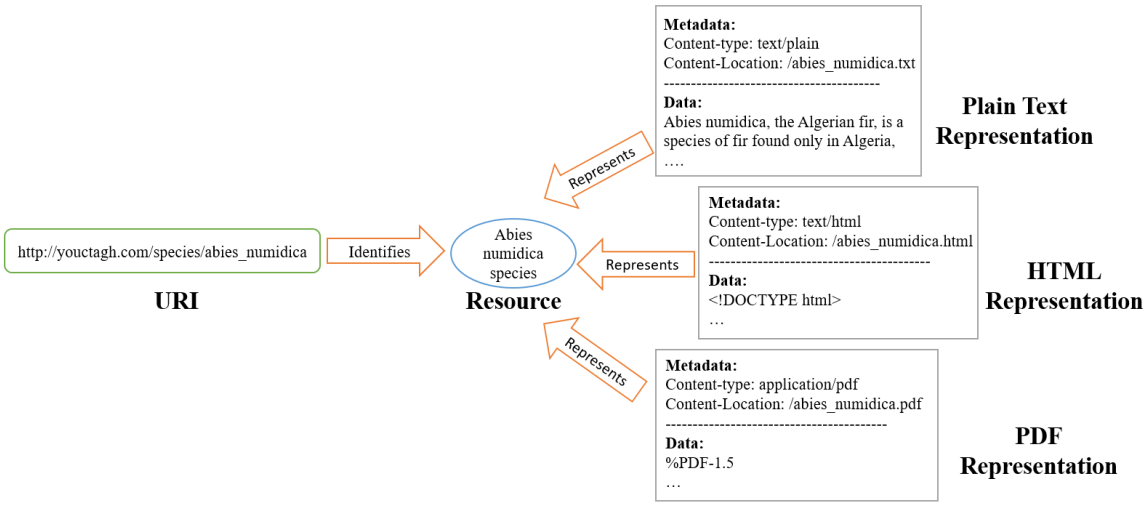



Figure 2.3: Representing a resource in different formats with multiple representations.

can reflect the state of the same resource. Nevertheless, the same URI could be used to identify the resource, thanks to the general architectural principles of the Web (which we will see in Section 2.1.4). The representations could for example differ in their format (cf. Story 2).

**Story 2.** *YoucTagh* wants to provide the same content already served in HTML (cf. Story 1) in PDF and Plain Text formats. However, he wants to use the same URI template for the species `http://youctagh.com/species/{species_name}`, while each representation has the appropriate data and metadata combination.

Figure 2.3 shows the new scenario applied to the *Abies numidica* resource. The current state of affairs raises several questions: (1) How can a client know if there are multiple representations of a resource? (2) How can a client formulate its preferences to request an appropriate representation? (3) How can the server select and provide the most appropriate representation? (4) What if no representation satisfies the client's needs, how should the server behave?

In the case of HTTP, the protocol provides the means to answer these questions, for example: (1) Using specific status codes, e.g. `300 Multiple Choices`, a server can indicate that multiple choices are available. (2) Using specific headers, e.g. `accept`, the client can express its format preferences. (3) Using a specific algorithm, the server can compare the client's preferences with the metadata of the available representations. (4) A server can either return a default representation or use specific

status codes, e.g. `4xx` or `5xx` to indicate the problem encountered. Thus, HTTP gives the ability to negotiate an appropriate representation of a resource when multiple representations are available. Further details on the practice of content negotiation in HTTP are discussed in Section 2.3 and in Chapter 3.

### 2.1.4 General Architecture Principles

In order for the Web to evolve and scale in a consistent way, some general architectural principles were applied to the three components discussed in the previous section (cf. Section 2.1) [88, Section 5]. First, *Orthogonality*, meaning that identification, interaction and representation evolve independently. This is achieved by allowing the introduced technologies to develop separately, even if one depends on the other. For instance, HTTP depends on URIs, yet they evolved independently as can be seen with specifications for URIs [18, 20] and HTTP [19, 58, 59, 61]. Another example is the variety of representation formats, e.g. HTML and PDF. Thanks to this principle, new formats are introduced without breaking existing applications. We have seen this aspect in Story 1, where we identify the resource *Abies numidica* without knowing the format of the served representation is HTML. Similarly, in Story 2 we have introduced new representations without changing the URI or modifying the first representation.

Second, *Extensibility*, which means that technologies are able to evolve without sacrificing interoperability, while maintaining backward compatibility. This can be seen in several Web technologies: URLs that locate resources, for example, allow access via FTP rather than just HTTP. HTTP also allows the retrieval of arbitrary file formats, not just HTML. In turn, HTML, through the `<a>` element, allows hyperlinking to any file format, not just HTML.[5]

Third, *Error Handling*, because agents communicate in the Web space, errors can occur for a variety of reasons (e.g. a HTML document is not well formed). Thus, Error handling is primordial, and can be achieved as follows: (1) by correcting the error if possible (e.g. by adding the missing HTML closing tags needed); (2) by recovering from the error by continuing the process, taking into account the fact that an error has occurred (e.g. by displaying the web page content up to the point where an error is found, and popping up an appropriate information message).

Finally, *Protocol-based Interoperability*, meaning that the Web defines its main concepts with protocols to drive interoperability, for example when specifying the syntax of the messages interchanged.

[5]Note that other elements such as `img`, `meta`, `script` and `video` also allow hyperlinking.

Our work also benefits from these general architectural principles as pointed out in later sections (see Chapter 5).

## 2.2 The Semantic Web

From the early days of the Web until the early 2000s, most of the content available on the Web was designed primarily for humans to read, rather than for machines to understand and manipulate. This was one of the points made by Tim Berners-Lee, James Hendler and Ora Lassila in their famous paper on the Semantic Web [21]. The Semantic Web, as its name suggests, aims at extending the Web with semantics, giving structure and meaning to web content so that software agents can consume and use it in a more informed and knowledgeable way. In what follows, we introduce the main concepts of interest to us, starting with the RDF data model used to describe resources and express meaning in Section 2.2.1. Then, we introduce RDFS and OWL, which allow the representation of rich and complex knowledge in Section 2.2.2. Finally, in order to be able to validate semantic content, we should undertake a process of semantic validation, which we explore in Section 2.2.3.

### 2.2.1 Resource Description Framework

The standard data model for describing resources on the Semantic Web is RDF [48]. RDF builds on the identification and linking structure of the Web. In RDF, IRIs are used to identify resources and the relationships that link them. RDF represents data as triples. An RDF triple takes the form of *(subject, predicate, object)*. The subject can be either an IRI or a blank node, the latter being useful for describing resources without giving them global identifiers. The predicate can only be an IRI, while the object can be either an IRI, a blank node or a literal. Literals are basic values that are not IRIs, for example strings, numbers or dates. A literal is associated with a datatype identified by an IRI to enable its parsing and correct interpretation, with the ability to use language tags if the literal has the datatype `http://www.w3.org/1999/02/22-rdf-syntax-ns#langString` to assert the used language. An RDF term is an IRI, a literal or a blank node. A set of RDF triples forms an RDF graph. Since IRIs are used to identify resources and relationships, if someone dereference them a content negotiation interaction can take place.

**Story 3.** *YoucTagh* wants to provide RDF representations of the resources offered by his server. Taking the resource *Abies numidica* as an example, he plans to use the currently available HTML content describing it as a starting point (cf. Listing 2.1).

The knowledge conveyed as HTML in Listing 2.1 could be represented by the RDF triples in Table 2.1.[6] The Figure 2.4 is a visual representation of such knowledge.

The RDF graph can then be serialised and served in one of the various serialisations available [153, Section 5]. Listing 2.2 shows an example of a common used serialisation named Turtle that we use throughout the thesis [138].

Turtle is a textual syntax that allows RDF graphs to be written in a compact form. It has many abbreviations for common usage patterns and datatypes. For example, we can see several in Listing 2.2:

- The `@base` declaration defines a base IRI such that *relative IRIs* such as `<species/abies_numidica>` can be resolved as: `<http://youctagh.com/species/abies_numidica>`
- The `@prefix` declaration defines a *prefix* that is used in *prefixed name*, for example, `rdf:type` that uses the prefix `rdf:`
- The `a` token is an abbreviation that represents the IRI: `http://www.w3.org/1999/02/22-rdf-syntax-ns#type`. It is equivalent to `rdf:type`
- The . and ; symbols are used to terminate *simple triples* and separate *predicate lists* respectively.
- `_:b1` defines a *blank node* with the *label* `b1`. Unlabelled blank nodes can also be used if needed using `[]`. For an example, check Listing 2.3
- `"meter"@en` defines an RDF literal with English language tag
- The `25` RDF literal is used as the equivalent of `"25"^^xsd:integer`

Listing 2.2: A Turtle representation of knowledge about *Abies numidica* using only RDF

```
1 @base <http://youctagh.com/> .
2 @prefix ex: <http://www.example.com/> .
3 @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4
5 <species/abies_numidica> rdf:type ex:Tree ;
6     ex:label "Abies numidica"@en;
7     ex:growsBetween  _:b1.
```

[6] In this example, `ex:label`, `ex:growsBetween`, `ex:minimum`, `ex:maximum`, `ex:unit`, and `ex:HeightRange` are IRIs, `_:b1` is a blank node, while "Abies numidica", "25", "35" and "meter" are literals.

```
_:b1    a    ex:HeightRange ;
             ex:minimum 25;
             ex:maximum 35;
             ex:unit "meter"@en.
```

| Subject | Predicate | Object |
|---|---|---|
| <species/abies_numidica> | rdf:type | ex:Tree |
| <species/abies_numidica> | ex:label | “Abies numidica”@en |
| <species/abies_numidica> | ex:growsBetween | _:b1 |
| _:b1 | rdf:type | ex:HeightRange |
| _:b1 | ex:minimum | ”25”^^xsd:integer |
| _:b1 | ex:maximum | ”35”^^xsd:integer |
| _:b1 | ex:unit | “meter”@en |

Table 2.1: RDF triples representing knowledge about *Abies numidica*

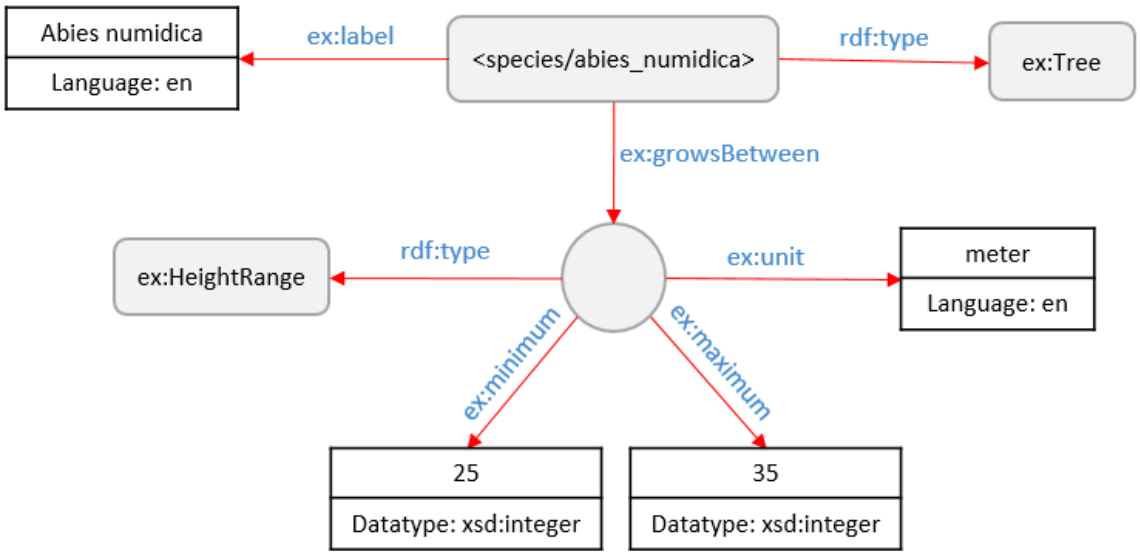



Figure 2.4: RDF graph describing knowledge about *Abies numidica*

In the remaining examples of the thesis, we omit the prefixes part of the listings. Table 2.2 provides the list of prefixes as well as their respective namespaces used throughout this manuscript.

| Prefix | Namespace |
|---|---|
| cnccmeans | https://w3id.org/cnco/ccmeans# |
| cnco | https://w3id.org/cnco# |
| cndimension | https://w3id.org/cnco/dimension# |
| cnprotocol | https://w3id.org/cnco/protocol# |
| cnstyle | https://w3id.org/cnco/style# |
| dbo | https://dbpedia.org/ontology/ |
| dbp | https://dbpedia.org/property/ |
| dct | http://purl.org/dc/terms/ |
| ex | http://example.com/ |
| foaf | http://xmlns.com/foaf/0.1# |
| ly | http://lyncis.com/ |
| npg | http://ns.nature.com/terms/ |
| owl | http://www.w3.org/2002/07/owl# |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| rdfs | http://www.w3.org/2000/01/rdf-schema# |
| sdo | https://schema.org/ |
| sh | http://www.w3.org/ns/shacl# |
| xsd | http://www.w3.org/2001/XMLSchema# |
| yt | http://youctagh.com/ |
| yto | http://youctagh.com/ontology/ |
| ytshape | http://youctagh.com/shapes/ |

Table 2.2: A summary of the list of prefixes used in this thesis.

To retrieve and manipulate data stored in such RDF graphs, the W3C recommendation for RDF query language SPARQL [154] may be used. Note that other query languages existed before [78].

### 2.2.2 RDF, RDFS and OWL

RDF provides the means to describe resources using RDF graphs. The RDF vocabulary [117] may be used to achieve this, by using RDF terms such as `rdf:type`, `rdf:Property`, `rdf:Statement`, where `rdf` is the common prefix used for the RDF vocabulary.[7] For instance, `rdf:type` may be used to model the instance relation.

However, triples by themselves do not ensure common semantics, which is why we need ontologies that provide a common agreement on the semantics of the concepts used. A widely accepted definition of an ontology is "an explicit specification of a conceptualization" [74], meaning that all users of an ontology commit to the same semantics of its concepts and their relations to each other.

An extension of the RDF vocabulary has been proposed called RDFS [32]. It allows the specialisation of classes through the use of the property `rdfs:subClassOf` as well as the specialisation of properties with `rdfs:subPropertyOf`, with `rdfs` being the common prefix used for the RDFS vocabulary. The ability to specify the domain and range of properties using `rdfs:domain` and `rdfs:range` respectively. The *Dublin Core Metadata Element Set* [33] is an example of a well-known vocabulary defined with RDFS.[8]

While RDFS alone might be sufficient in some situations, its expressiveness is not enough for complex usage. For example, one cannot restrict the possible values of a property or define co-references. Such semantic modelling is provided by OWL [73].[9]

OWL [120, 73] is a semantic markup language. It is based on RDF and RDFS to enable the definition and modelling of complex and rich knowledge. It is based on description logics [86] and has several sublanguages (called profiles) with different features and properties [123].

Figure 2.5 presents a set of concepts and properties of a simple tree ontology. Listing 2.3 shows a Turtle representation of the knowledge about *Abies numidica* using this ontology.

[7] The RDF vocabulary could be found at: `https://www.w3.org/1999/02/22-rdf-syntax-ns#`

[8] Another example is the Friend Of A Friend (FOAF) vocabulary before it was rewritten in OWL. An old snapshot of a version using only RDFS can be found at: `https://web.archive.org/web/20020802154059/http://xmlns.com/foaf/0.1/`

[9] `owl:allValuesFrom` and `owl:sameAs` properties could be used for such purposes. More on identity management and equivalence relationships is covered in Section 7.1.

Listing 2.3: A Turtle representation of the knowledge about *Abies numidica* using the ontology presented in Figure 2.5

```
<species/abies_numidica> a yto:Tree;
    yto:label "Abies numidica"@en;
    yto:growsBetween [
        a yto:HeightRange;
        yto:minimum 25;
        yto:maximum 35;
        yto:unit "meter"@en;
].
```

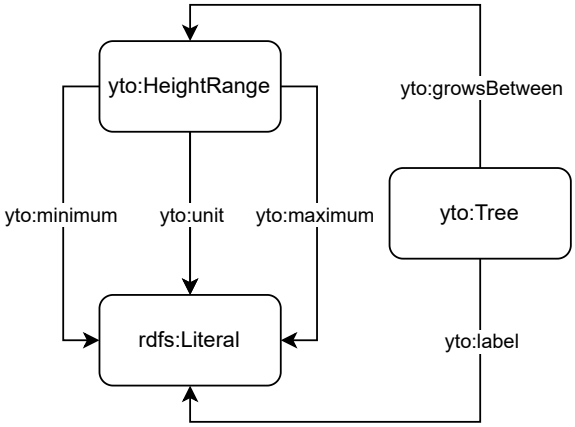



Figure 2.5: A simple tree ontology to describe knowledge about *Abies numidica*

### 2.2.3 Semantic Validation

When exchanging data on the Web, agents may have expectations about what acceptable data looks like. To achieve this, constraint languages are essential. Over the years, just like the data definition language for relational databases, several constraint languages have been proposed for different purposes: *XML Schema* [66], *REgular LAnguage for XML Next Generation* [179] or *Schematron* [180] for *Extensible Markup Language (XML)* [31], *JSON Schema* [185] for *JavaScript Object Notation (JSON)* [30], and *Concise Data Definition Language* [26] for *Concise Binary Object Representation (CBOR)* [28].

RDFS and OWL are designed to enable inference, not validation. For example, if we have the triple (`<bob>`, `foaf:knows`, `<alice>`) we can infer that both `<bob>` and `<alice>` are of type `foaf:Person` because the specification defines the domain and range of this property as `foaf:Person`.[10] The same can be said for OWL restrictions, which are not actually data constraints. For example, suppose an application has an `owl:maxCardinality` 1 restriction, which states that a `foaf:Person` can have zero or one value for the `ex:bestFriend` property. If we have an instance of `foaf:Person`, say `<alice>`, and that person has two best friends `<bob>` and `<rick>`, an OWL processor will not raise an error, however it will try to find a setting in which such a scenario is possible. Consequently, an OWL processor will infer that `<bob>` and `<rick>` must actually be the same person, just with different URIs. This is possible

[10] Specification of the `foaf:knows` property http://xmlns.com/foaf/0.1/#term_knows

because OWL does not assume unique names[11] (more on this topic in Section 7.1).

Just as important as inference, data validation is critical in many use cases.[12] For example, the use case to define RDF graph expectations and patterns that are accepted/forbidden in an RDF graph. Also, the one to determine whether or not a given graph is valid when consumed (by a client) or produced (by a server).

**Story 4.** *YoucTagh* wants all the served RDF representations to respect and conform to a schema he has in mind. For example, every tree must have a label and a height growth range. He also wants to use such a schema to prevent incorrect data from being included, e.g. the height of a tree cannot be a negative integer.

Validation of RDF data went through several stages [67, Chapter 3.3]: SPARQL ASK or SELECT queries that capture the illegal patterns in a valid RDF graph; open services for lifecycle collaboration resource shape [148]; SPARQL Inferencing Notation (SPIN) [96]; and ShEx [137]. Finally, on 2017 Shapes Constraint Language (SHACL) became the W3C recommendation for such purpose [97]. Listing 2.4 provides a snippet of the SHACL shape graph of the requirements expressed in Story 4: all instances of type `yto:Tree` must:

1. have at least one path `yto:label`;
2. have at least one path `yto:growsBetween`. The value node of `yto:growsBetween` must be of type `yto:HeightRange` and must validate three shapes:
   (a) `ytshape:MinRangeShape` indicating that the value of the range following the path `yto:minimum` must be positive;
   (b) `ytshape:MaxRangeShape` indicating that the value of the range following the path `yto:maximum` must not exceed a certain value;
   (c) `ytshape:MLMRangeShape` indicating that the minimum value of the range must not exceed the maximum value of the range.

Listing 2.4: A snippet of the SHACL shape graph of the requirement expressed in Story 4

```
1 ytshape:TreeShape a sh:NodeShape;
2   sh:targetClass yto:Tree; # all instances of type yto:Tree
```

[11] The absence of unique name assumption implies that two different URIs could be used to refer to the same resource.

[12] See for example the SHACL Use Cases and Requirements document: https://www.w3.org/TR/shacl-ucr/

```
3   sh:property [
4    sh:path yto:label; # must have at least
5    sh:minCount 1;];   # one path yto:label
6   sh:property [
7    sh:path yto:growsBetween; # must have at least
8    sh:minCount 1;            # one path yto:growsBetween
9    sh:class yto:HeightRange; # the value node must be
10                             # of type yto:HeightRange
11   sh:property ytshape:MinRangeShape; # must validate
12   sh:property ytshape:MaxRangeShape;  # these three
13   sh:property ytshape:MLMRangeShape;]. # shapes
14  ...
15  ytshape:MinRangeShape a sh:PropertyShape;
16   sh:path yto:minimum;     # the value of yto:minimum
17   sh:minExclusive 0.       # must be positive
```

## 2.3 Content Negotiation in HTTP

In the previous sections we saw that the three main components of the Web architecture are: identification, interaction and representation. More practically, resources are identified through the use of URIs, and protocols such as HTTP, allow the request and delivery of resource's state (cf. Story 1 for an example). This section shows how such interactions are possible in HTTP, starting with a simple request/response in Section 2.3.1, and extending it with negotiation in Section 2.3.2.

### 2.3.1 Basic HTTP Interaction

HTTP defines several *methods* to enable interactions, each with its dedicated semantics [61, Section 9]. `GET` is used to retrieve a representation of the resource identified by a URI. `HEAD` is identical to `GET` except that `HEAD` is used to retrieve metadata only, so no content needs to be sent in the response. `POST` is used to send data from the client to the server. Listing 2.5 and Listing 2.6 show a request/response example where a client requests a representation of the resource *Abies numidica* and the server responds with the HTML representation it has (cf. Story 1).

Listing 2.5: A client requesting a representation of the *Abies numidica* resource

```
1  GET /species/abies_numidica HTTP/1.1
2  Host: youctagh.com
```

Listing 2.6: The server responding to a `GET` request with an HTML representation of the *Abies numidica* resource

```
1  HTTP/1.1 200 OK
2  Date: Mon, 31 Jul 2023 09:45:09 GMT
3  Content-Location: abies_numidica.html
4  Content-Type: text/html
5  Content-Length: 305
6
7  <!DOCTYPE html>
8  <html lang="en">
9    <head> <title>Abies numidica</title> </head>
10   <body>
11     <a href="http://youctagh.com/species/abies_numidica"> Abies
          numidica </a>
12     is a
13     <a href="http://youctagh.com/plant/tree"> tree </a>
14     growing between 25 and 35 meters.
15   </body>
16 </html>
```

This is a simple client/server interaction where a resource has only one representation. However, as we saw in the previous chapter, a resource identified by a URI can have multiple representations (cf. Story 2). The representations may differ, for example, in the format or language used. To enable the selection of an appropriate representation from several available ones, we use the content negotiation mechanism.

### 2.3.2 HTTP Interaction with Negotiation

HTTP comes with a number of concepts that collectively contribute to the successful functioning of the content negotiation mechanism. The following is a brief overview of the core concepts, which we will discuss in more detail in Section 3.3.

First, HTTP allows negotiation in multiple *content negotiation styles* (or patterns). The original HTTP protocol specification suggested two styles: *preemptive* (currently named *proactive*), in which the server is responsible for selecting the representation to provide based on some requested preferences, and *reactive*, in which the server provides a list of available representation alternatives for the requester to choose from [16]. Later, other styles were proposed, each with its pros and cons, which we will discuss in much detail in Section 3.3.2. The *HTTP message* consists of three parts in the following order:

- A single line describing a sent request containing the method used, or a received response with its corresponding status. See for example the first line of Listing 2.5 and 2.6.

- An optional set of HTTP headers (colloquially *headers*). *Headers* are name/-value pairs that provide a way to exchange request/response descriptions. See for example line 2 in Listing 2.5 and lines $2-5$ in Listing 2.6.

- An optional message body. Note that for example the first Listing 2.5 does not contain this part because it is a `GET` request. In contrast, in the Listing 2.6 of the response, the body starts at line 7.

The HTTP method, response status code and headers collectively contribute to the semantics of an interaction [61, Section 6.4.1]. Some header examples are `accept-language`, which is used as a means of conveying the preferred set of natural languages of a representation. Or, `vary` which indicates the headers that, when changed, may lead to a different representation [61, Section 12.5]. Some status codes include `200 OK`, which indicates that the request was successful, `300 Multiple Choices`, which indicates that the requested resource has more than one representation, or `406 Not Acceptable` which indicates that no acceptable representation could be found considering the given preferences. We provide more details on these concepts in the subsequent sections.

Going back to Story 2, a client wishing to negotiate a plain text representation can send a request as shown in Listing 2.7. The server will select the appropriate representation and send back a response as shown in Listing 2.8.

Listing 2.7: A client requesting a plain text representation of the *Abies numidica* resource

```
1 GET /species/abies_numidica HTTP/1.1
2 Host: youctagh.com
3 accept: text/plain
```

Listing 2.8: The server select and serve the appropriate representation of the *Abies numidica* resource

```
1 HTTP/1.1 200 OK
2 Date: Mon, 31 Jul 2023 14:02:52 GMT
3 Content-Location: abies_numidica.txt
4 Content-Type: text/plain
5 Vary: accept
6 Content-Length: 80
7
8 Abies numidica, the Algerian fir, is a species of fir found only
    in Algeria, ...
```

## 2.4 Conclusion

In this chapter, we illustrated the problem of content negotiation as well as the different concepts related to it. We discussed the main concepts and principles underlying the Web architecture, as well as its extension, the Semantic Web, and the essential building blocks used in our research.

In Section 2.1 we gave an overview of the core design components of the Web, namely identification, interaction, and representation, as well as the general architectural principles that apply to all of these architectural bases, i.e. orthogonality, extensibility, error handling, and protocol-based interoperability.

In Section 2.2 we introduced the main concepts of interest to us that are part of the Semantic Web. Starting with the RDF data model, which is used to describe resources. Then RDFS and OWL, which allow the representation of rich and complex knowledge. We conclude this section with semantic validation techniques, which are of great help in validating and checking the conformity of semantic content to some desired constraints and rules.

In Section 2.3 we defined content negotiation and how it was introduced and used in the HTTP protocol. We showed how client/server interactions are possible in HTTP, starting with a simple request/response interaction, and then extending it with negotiation to ask for a resource representation when several are available.

In the next chapter we provide a thorough state of the art on the semantic content negotiation problem.

Chapter 3

# Semantic Content Negotiation

## Contents



In the previous chapter we discussed the Web and its architecture as well as its extension the Semantic Web. In this chapter we continue our investigation, but focus on the practice of content negotiation and discuss it from several angles. The purpose of this chapter is to better identify and understand multiple limitations that motivate our work.

First, in Section 3.1, we present the different stages of content negotiation, emphasising that it is not a monolithic process, but rather a multi-layered one. Second, in Section 3.2, we examine attempts to formalise it (cf. Limitation 1). Later, in Section 3.3, we survey the characteristics of content negotiation, presenting the different content negotiation dimensions, styles, as well as ways to convey constraints to be considered in the process (cf. Limitation 3). Finally, in Section 3.4, we expand on the profile dimension, which allows for the negotiation of semantics (cf. Limitation 2).

## 3.1 Content Negotiation Stages

Various works in the literature have proposed a description or a framework of the content negotiation process, for example as part of the HTTP specification [61, Section 12.5], or in a number of different RFCs, such as the use of content negotiation in messaging services [93, Section 3], the proposal of a protocol-independent content negotiation framework [92, Section 3], or also when introducing the transparent content negotiation style [84, Section 4], in addition to contributions using content negotiation as a means to achieve other desirable objectives, such as having an effective adaptation service in mobile computing [115, Figure 2]. Taken together, these works support the view that the content negotiation process is not monolithic, but consists of several stages that together contribute to the success of the mechanism.

Decomposing the content negotiation process is advantageous in our work, as one can identify a particular stage of interest and improve that part of the process. More importantly, this way we also comply with different principles of the Web (cf. Section 2.1.4), for example *orthogonality*, meaning that each stage is orthogonal to the others, so that we can improve one stage separately without considering the others (the *extensibility* principle).

In our work we consider five distinct stages. Next, we present each stage and illustrate it with parts of the stories presented in the previous sections, Figure 2.3 and listings 2.7 and 2.8:

**Discovery:** In the first stage, the focus is on answering questions such as: How does a client discover a negotiable resource on the Web space? and, at the same time, how does a server announce the availability of said resource?

For example, the client discovers the resource *Abies numidica* identified by the URI `<http://youctagh.com/species/abies_numidica>`.

**Request Formulation:** At this stage, a client communicates to the server its request for a resource, in addition to any constraints and preferences.

For example, as shown in Listing 2.7, the client formulates its request for the resource *Abies numidica* and specifies its preference for a plain text representation via the `Accept` header.

**Selection/Adaptation:** Next, at this stage, the server selects an alternative representation i.e. a variant, and optionally adapts it to the client's constraints and preferences.

Following the same example, the server selects the plain text representation from the available ones (cf. Figure 2.3).

**Response Indication:** After that, the server constructs its response containing the chosen representation(s) with the appropriate metadata, preferably indicating the reason for that choice. In the case of a negative response, alternatives are proposed if possible. This stage ends by sending the response back to the client.

Listing 2.8 shows a positive response indicated by the `200 OK`, the reason is expressed in the `Vary` header and metadata, for example with `Content-Location` and `Content-Type`.

**Response Interpretation:** In the final stage, the client interprets the received response and either continues if necessary, e.g. if alternatives are proposed, or terminates the negotiation process (by accepting the server's reply).

In our example, we assume that the client accepted the provided representation.

In our efforts to enhance the content negotiation process in order to allow for relaxed and flexible negotiation, we will be guided by these stages, i.e. we will improve the negotiation at each stage. In Chapter 5 we detail these improvements and map them to their respective stages.

## 3.2 Content Negotiation Formalisation

The HTTP specification uses Augmented Backus-Naur Form notation to define the syntax and derivation rules [61, Appendix A]. However, apart from the syntax for the header fields, no abstract model or formalisation is provided to specify the content negotiation process. RFC 2703 proposes a protocol-independent content negotiation framework. It presents the terminology, abstract framework, and goals for such a framework. However, as explicitly stated, "The abstract framework does not attempt to specify the content negotiation process" [92].

Lerouge formalised the content negotiation problem in his PhD thesis [111, Section 4.4]. However, his work was done in the specific context of describing the negotiation process of scalable bitstreams. The aim was as follows: having a bitstream that can be split into a sequence of parcels, derive an adapted bitstream from the original one, such that each adapted parcel satisfies all the constraints imposed by a given usage environment, and at the same time maximises a given quality function. For this purpose, the author defined an abstract model for scalable bitstreams and

the content negotiation problem in this context. As a consequence, the formalisation does not take into account the concepts of web server, request or response. Furthermore, the abstract model is not generic enough to reflect current practices and naming conventions on the Web, e.g. resource instead of bitstream and package, or representation instead of version of a package. Such genericity is needed in our work to reflect the current practices of the HTTP protocol to do content negotiation with its predefined dimensions [61, Section 12.5]. In addition, such a generic model should be extendable to the Semantic Web to achieve semantic content negotiation using semantic validation languages (cf. Limitation 1).

In the body of literature there are different definitions and formal descriptions of different semantic validation languages, such as ShEx [139], JSON schema [10] and XML schema [35]. A formal definition of SHACL was also considered as part of the work to propose a mapping from a relational database to SHACL [174], itself based on an earlier work [47] proposing an abstract syntax and formal semantics for SHACL. However, if we choose SHACL as a semantic validation language, these contributions do not include the notion of shape severity, which is of great value to allow a better and finer definition of constraints, and thus negotiation of content.

In the same way, if we turn to ontologies and try to find a specification for content negotiation, we can find the HTTP vocabulary [98], where the core concepts of the specification are present, e.g. `Message`, `Request` and `Response`, but it also falls short in providing the means to specify the content negotiation dimension and style for example. An extension to this vocabulary has been proposed to describe HTTP interactions [112], but it suffers from the same shortcomings. The lack of ability to describe the content negotiation characteristics of a content provider can also be observed in other relevant vocabularies, e.g. Hydra [103], the profiles vocabulary [8], the RDF presentation ontology [107]. The linked data API vocabulary [173] defines some content negotiation related concepts such as `contentNegotiation`, `ContentNegotiationStrategy`, `suffixBased`, `parameterBased` or `mimeType` and specific media type to be negotiated. Still it is not generic for other dimensions and does not specify the content negotiation style used. Defining an ontology that takes these characteristics into account would be beneficial for enabling software agents to autonomously engage in content negotiation with servers (Limitation 5): A software agent would consume a server description of its content negotiation capabilities as an RDF document and engage in the process accordingly.

## 3.3 Content Negotiation Characteristics

In the HTTP specification, when defining content negotiation [61, Section 12], three characteristics emerge:

- first, the content negotiation *dimension* over which the available representations may vary;
- second, the content negotiation *style*, i.e. the pattern or manner by which the content negotiation process is carried out;
- third, the content negotiation *constraint conveyance means*, i.e. the means of conveying constraints about the different content negotiation dimensions.

While the specification illustrates constraint conveyance with headers, note that it relies on URI for resource identification [61, Section 4], and the URI definition allows for queries, which are used as another means of conveying preferences. [20, Section 3.4].

In the following subsections, we detail these characteristics, starting with content negotiation dimension in Section 3.3.1, then content negotiation style in Section 3.3.2, followed by content negotiation constraint conveyance means in Section 3.3.3. To conclude this section, we show how we can benchmark existing contributions that employ content negotiation using these characteristics (Section 3.3.4).

### 3.3.1 Content Negotiation Dimension

A content negotiation dimension, denotes a type of constraint or preference that the client wants the server to take into account when processing the request. The content negotiation dimensions are the primary distinguishing factors between a set of alternatives. Two alternative representations vary according to one or multiple content negotiation dimensions, for which we give an overview with their respective definitions and their usage. Note that some of the following dimensions have been present since the first HTTP protocol specification; others have emerged over time and are actively used; some are ad-hoc dimensions used for specific use cases. The Appendix B.1 provides illustrative examples of different representations of each dimension.

**Media type** Formerly known as MIME, this dimension defines the preference for the format of the content to be delivered, e.g. *HTML*, *PDF* or *plain text*. Clients

can express a preference for media types[1] in the HTTP protocol using the `accept` header [61, Section 12.5.1], for example:

```
accept: text/html, application/pdf, text/plain
```

**Charset** If the request concerns textual content, the client can negotiate the encoding of the characters it supports or prefers. Note that this dimension was more commonly used before *UTF-8* became widely supported. In the HTTP protocol, the client can express a preference for character sets[2] using the `accept-charset` header [61, Section 12.5.2], for example:

```
accept-charset: iso-8859-5
```

**Encoding** Content encodings[3] can be applied to a representation, i.e. an encoding transformation to allow, for instance, compression. In the HTTP protocol, the user can express such a preference by using the `accept-encoding` header [61, Section 12.5.3], for example:

```
accept-encoding: compress, gzip
```

**Language** Different end users from different linguistic areas are most likely to prioritise content in different natural languages. This dimension allows clients to specify the set of natural languages that are preferred in the response. In the HTTP protocol, the user can express a preference for a set of languages[4] using the `accept-language` header [61, Section 12.5.4], for example:

```
accept-language: en, fr, ar
```

[1] List of media types registered in IANA: https://www.iana.org/assignments/media-types/media-types.xhtml

[2] List of character sets registered in IANA: https://www.iana.org/assignments/character-sets/character-sets.xhtml

[3] List of registered content encodings in IANA: https://www.iana.org/assignments/http-parameters/http-parameters.xhtml

[4] List of language subtags registered in IANA: https://www.iana.org/assignments/language-subtag-registry/language-subtag-registry

**Feature** Shortly after proposing the standard for the Hypertext Transfer Protocol [60], the IETF *Content Negotiation (conneg) Working Group* (from 02/1998 to 10/2000) was formed,[5] Its main objective was to explore the negotiation of document presentation elements that are not captured naturally by the media type, that the group later called *features*. Feature negotiation was defined to permit negotiation in areas not covered by the type, charset, and language dimensions [84, Section 4.8]. Some examples are: colour capabilities of the user agent, screen size, output medium (e.g. screen or paper). A major advantage of this new dimension is that new ones (which can be seen as sub-dimensions of the feature dimension) can be added without the need for new standardisation efforts, but simply by registering a new *feature tag* [85, Section 3.3]. A client can express its preferences[6] in the feature dimension using the `accept-features` header, for example:

```
accept-features:  colordepth=5, paper = A4
```

**Capability** With the introduction of RDF and the proliferation of new wireless devices in the early 2000s, two main vocabularies have been proposed to describe device capabilities and user preferences, namely: Composite Capability/Preference Profile (CC/PP) [94] by W3C and User Agent Profile (UAProf) [132] by Wireless Application Protocol (WAP) Forum. These vocabularies define the capabilities and preferences of a receiver's hardware or software[7]. For instance:

**colour** to state if the receiver can handle colour, greyscale or only black and white;

**screen size** to define the supported height and width of the screen.

This is often referred to as the delivery context of a device and can be used to guide the negotiation and adaptation of content presented to that device. One may notice some similarities between CC/PP and the media properties described earlier, or CC/PP and UAProf. In fact, the small common vocabulary defined for media features [119] has been used as the basis for the CC/PP client common vocabulary.[8]

[5] An archived version of the working group website: https://web.archive.org/web/20130308181729/http://www.imc.org/ietf-medfree/.

[6] Media Feature Tags: https://www.iana.org/assignments/media-feature-tags/media-feature-tags.xhtml

[7] An example of a Nokia CC/PP file https://web.archive.org/web/20211024131258/http://nds.nokia.com/uaprof/N601Tr100.xml

[8] Note that CC/PP lacks a matching mechanism as defined in the media feature framework in terms of comparisons with fixed values (e.g. 'pix-x¡=640') and attribute values that occur in some combinations (e.g. 'pix-x=640' AND 'pix-y=480' OR 'pix-x=800' AND 'pix-y=600') [94, Section E.1 ].

Also, as explicitly mentioned in the CC/PP specification, any valid UAProf profile is also a valid CC/PP profile. A client can convey its CC/PP profiles using the `profile` header,[9] as explained in the CC/PP exchange protocol based on HTTP extension framework [131], for example:

```
profile:  "http://www.path.to/profile"
```

**Time** Some resources on the Web evolve over time, and so do their representations. To facilitate the retrieval of prior states of a resource, negotiation in the time dimension is required. The HTTP-based *Memento* framework [159] proposes a datetime negotiation to do just that. Archiving is a use case where Memento could be of great benefit. The client can express the preference in this dimension using the `accept-datetime` header, the value of the header is expressed according to the RFC1123 format [29], for example:

```
accept-datetime:  Sun, 01 Nov 2020 09:00:00 GMT
```

**Version** A resource may be exposed on the Web through an API, and content negotiation may be supported to provide different representations [76]. Such APIs may change over time, so API versioning is required [77]. In HTTP, the client can use the `accept-version` to indicate the desired version. This is the suggested way in the blog [90] or also when developing REST APIs with *Node.js* [51, Chapter 5]. Similarly, the GitHub API documentation mentions that versioning is handled with the `X-GitHub-Api-Version`.[10] Note that semantic versioning can be used when assigning versions,[11] for instance:

```
accept-version:  2.0.1
```

**Coordinate Reference System** Several coordinate reference system can be used to locate objects on Earth. As a result, different representations of the same object may exist in different systems, e.g. *WGS84* or *ETRS89*. A client can use the HTTP header `accept-crs` [1] to express the preferences in this dimension as as proposed in

[9] Not to be confused with `accept-profile` that will be presented later.

[10] GitHub Docs - API Versions: https://docs.github.com/en/rest/overview/api-versions?apiVersion=2022-11-28

[11] Semantic Versioning: https://semver.org/

the Open Geospatial Consortium (OGC) & W3C best practices document on spatial data on the Web [171, Section 13.1], for example:

```
accept-crs:  EPSG:4326
```

**Formatting** Formatting generally refers to the way in which a document (text, image, etc.) is organised, such as the minified version of a `JavaScript` file or the citation style of a bibliography. For example, Crossref, DataCite and mEDRA support negotiation of formatted citations [187] via the `text/x-bibliography` content type and the `style` parameter, which can be used to select from the list of citation style names found in the citation style language repository.[12] The following example requests the citation to be formatted using the Modern Language Association style:

```
accept:  text/x-bibliography; style=modern-language-association
```

**Presentation** A significant part of the content consumed, produced or published on the Web is not presented in RDF [104]. This observation is particularly true in the Web of Thing (WOT) domain, where HTTP servers and clients prefer to exchange lightweight documents. Consequently, mechanisms for lifting, i.e. decoding a typed octet stream into an RDF graph (e.g. RDF Mapping language (RML) mappings [49], XSPARQL [4], SPARQL-Generate [106]), and lowering, i.e. encoding an RDF graph into a typed octet stream (e.g. XSPARQL [4] and SPARQL-Template [45]) have been proposed. An RDF presentation is a combination of lifting, lowering and validation rules (e.g. SHACL [97] or ShEx [137]), and representation validation rules (e.g. JSON Schema [185] and XML Schema [35]). These rules would allow a coherent interpretation for a representation as RDF (lift), validation of the RDF graph, and the ability to generate the representation back from the RDF graph (lower). Thus, a client might express its preference for an RDF presentation using the `accept-presentation` header, the value of that field is an absolute IRI identifying an RDF presentation that can be described using the RDF presentation ontology [107], for example:

```
accept-presentation:  https://w3id.org/rdfp/example/graph
```

[12]The citation style language repository: https://github.com/citation-style-language/styles

**Vocabulary** Resources on the Web can be described using RDF and then serialised in various formats. However, in addition to negotiating the serialisation of the graph to be delivered, a client may want an adapted graph, for example by omitting triples with predicates that the client does not understand, or by providing inferred triples in case the client is unable to use the base triples. A W3C wiki is available that summarises a discussion on extending HTTP for vocabulary negotiation [11]. For this reason, `accept-vocabulary` has been proposed to convey such vocabulary preferences. For instance:

```
accept-vocabulary:  http://schema.org
```

**Schema** Resources can be described in different ways when using XML or RDF. For example, an XML document may use some XML schemas to encode content. In this case, a client may have a preference for one schema over another when making its request, and may wish to express how the content in the response should be structured [160]. One solution that has been suggested is to use the `accept-schema` header. Such as:

```
accept-schema:  http://example.com/schema/schema-2
```

**Profile** Similarities can be observed between the dimensions previously described, e.g. *feature* and *capability*, also *formatting* and *presentation* or even *vocabulary* and *schema*. What these dimensions have in common is that they attempt to describe the structure or semantic preferences that the response must conform to. To reflect this definition and to be general enough for future extensions, the DXWG proposed the profile vocabulary [8] as well as how to achieve content negotiation in the profile dimension [161]. In parallel, an IETF work is exploring ways to indicate, discover, negotiate and write profiled representations [163]. A profile to them denotes "a description of structural and/or semantic constraints on resource representations that apply in addition to the constraints inherently indicated by their media type" [163]. Since our research focus is on *semantic* content negotiation, this is the main dimension we will explore. More details on this dimension will be presented in Section 3.4. A client can express its preference in this dimension using the `accept-profile` header, for example:

```
accept-profile:  http://example.org/shapes/shape-1
```

Other dimensions have been identified from use cases, but do not have a corresponding concrete proposal.

**Accuracy** Content that describes physical quantities may use different measurement units that provide a standard by which the numbers used in a measurement can be properly interpreted and/or mapped to other units. A client may want to request a representation that meets a particular precision or unit of measurement, e.g. *ucum* [151], in order to be able to consume and use it. Such a use case is discussed in the DXWG.[13]

**Authorisation** Content on the Web may be subject to access control policies. As a result, access to some resource representations may be restricted, e.g. for privacy reasons. A client may want to request a representation only if it complies with certain regulations and if they have the appropriate permissions. Note that when the content negotiation profile is used with SHACL, approaches such as SHACL-ACL [145] could be used as a starting point.

**Entailment regime** Using an entailment regime, additional RDF statements can be inferred from explicitly given assertions. A client may wish to request RDF content that conforms to a shape, but after using a particular entailment regime, e.g. RDFS [71]. Note that when the content negotiation profile is used with SHACL, the `sh:entailment` property can be used to satisfy such a need.

**OWL Profile** An OWL profile is a reduced version of OWL that trades some expressiveness for reasoning efficiency. Three profiles are specifies by the OWL profiles specification [123]: OWL 2 EL, OWL 2 QL and OWL 2 RL. A client may wish to request an ontology in a particular OWL profile.[14]

**Summary** The abundance and length of content on the Web is making it increasingly difficult for humans to assimilate and use. For this reason, a user may wish to negotiate a summary of content rather than the full version with some specific characteristics such as number of words or number of paragraphs. A similar use case for datasets is discussed in the DXWG.[15]

Table 3.1 summarises the content negotiation dimensions with some references to existing usage examples.

[13] Modelling data precision and accuracy use case: https://www.w3.org/TR/dcat-ucr/#ID15

[14] Note that it is rare for an ontology to be provided in multiple OWL profiles. For an example see: http://purl.org/az/List

[15] Summarisation/Characterisation of datasets: https://www.w3.org/TR/dcat-ucr/#ID33

| Dimension | Ref [16] | Date | Type | Usage | Header |
|---|---|---|---|---|---|
| Media type | [15] | 1992 | RFC | [87, 36, 54] | `accept` |
| Charset | [19] | 1996 | RFC | [36, 54] | `accept-charset` |
| Encoding | [15] | 1992 | RFC | [36, 54] | `accept-encoding` |
| Language | [15] | 1992 | RFC | [36, 54] | `accept-language` |
| Feature | [84] | 1998 | RFC Note | [133] | `accept-feature` |
| Capability | [131] | 1999 | W3C | [36, 54] | `profile` |
| Time | [159] | 2013 | RFC | [176, 70] | `accept-datetime` |
| Version | [90] | 2022 | Blog Article | [190] | `accept-version` |
| CRS | [1] | 2019 | RFC Draft | [188] | `accept-crs` |
| Formatting | [187] | 2022 | Blog Article | [198] | `accept` + `style` parameter |
| Presentation | [104] | 2018 | Article | [105] | `accept-presentation` |
| Vocabulary | [11] | 2006 | W3C Wiki | [72] | `accept-vocabulary` |
| Schema | [160] | 2016 | Article | [142] | `accept-schema` |
| Profile | [163] | 2021 | IETF Draft | [40, 39, 38, 41] | `accept-profile` |

Table 3.1: A summary of the content negotiation dimensions with some references for usage examples.

### 3.3.2 Content Negotiation Style

A content negotiation style or *protocol level pattern*, denotes a particular pattern by which the content negotiation process is carried out and can be made visible within the HTTP protocol or other protocols that support content negotiation. Six major content negotiation styles have emerged since the proposal of the Web and the first HTTP discussions: *proactive*, *reactive*, *request*, *transparent*, *active* and *conditional*. Note that these styles are not mutually exclusive, meaning that they can be combined to achieve a desired result given that each has its trade-offs in applicability and practicality. Next, we give an overview of the various content negotiation styles with their respective definitions and usage, in addition to pointing out the advantages and disadvantages associated with using a given style. Appendix B.2 provides illustrative examples of different request/response interactions in various styles.

[16] A reference introducing or describing this dimension, though not necessarily the first which does.

**Proactive** Also called *preemptive* or *server-side*, because in this style the preferences are sent by the client/user agent in the request (cf. Section 3.3.3), and thus the server is encouraged to select its preferred representation using a selection algorithm (see for example the Apache negotiation algorithm [191], and the formula for calculating the overall quality score of a representation[17]) [61, Section 12.1]. The choice is made by considering the possible representations according to the information provided in the request (cf. Section 3.3.1). In addition to this explicit information (e.g. in the `accept-*` headers), there is implicit information such as the network address of the client and content of the `user-agent` header. However, note that sometimes these implicit characteristics are not accurate. This is humorously told in a blog post about the history of the `user-agent` header.[18]

Any content in the response can be affected by the conveyed preferences, including resource representations and error representations. The `vary` header [61, Section 12.5.5] is usually sent in a proactive style response to indicate which content negotiation dimensions were considered in the selection algorithm. Some status codes of interest in this style are `200 OK`, `303 See Other`, `404 Not Found` and `406 Not Acceptable`. `200 OK` is used when a representation is provided either by taking into account the preferences or when providing a default is judged better than no response [61, Section 15.3.1]. `303 See Other` is convenient to be used as described in *Cool URIs* for the Semantic Web [150] specification to redirect to the correct representation. `404 Not Found` usually indicates that no representation could be found by the server [61, Section 15.3.1]. This status code is different from `406 Not Acceptable`, which indicates that none of the currently available representations would be acceptable to the client/user agent, meaning that preferences have been taken into account.

This style is advantageous to minimise the amount of back and forth interaction, as preferences are sent up front to the server, which runs its algorithm and sends back a response. Also, the client does not need a description of the selection algorithm to make a choice, as this is done by the server. However, there are some drawbacks to using this pattern. First, for some users, it is not always possible for the server to determine exactly what might be *best* for them, since this would require full knowledge of both the capabilities of the user agent and the intended use of the response. Second, sending the preferences in every request can be inefficient (since not all resources are subject to content negotiation) and could lead to potential problems (see for example product information disclosure [61, Section 17.12] and browser fingerprinting [61, Section 17.13] issues). Third, especially if a server handles multi-

[17] An example of a formula for calculating a representation score [84, Section-19.1]

[18] https://webaim.org/blog/user-agent-string-history/

ple content negotiation dimensions, algorithms for selecting/generating responses to a given request may have high complexity. Finally, since responses are tailored to specific preferences, this would limit the use of shared caching.

**Reactive** Also called *agent-driven* or *client-driven*, because in this style the final selection of content is made by the user/user agent and not by the server i.e. when the server receives a request it sends back a list of alternatives to choose from. The server does not choose a representation to deliver, either because it is incapable of proactive content negotiation, or because it does not have enough knowledge to complete the selection of the representation with the given preferences, or because it is configured to do so. The user agent can make the choice on behalf of the user if it has sufficient knowledge of the user's preferences or it has a default behaviour. Otherwise, it displays the list of links to the alternative representations so that the user can make the final choice [61, Section 12.2].

Some status codes of interest in this style are `300 Multiple Choices` and `406 Not Acceptable`, both being sent with a list of alternatives to choose from. The former is used to indicate that reactive content negotiation is preferred over proactive, the latter indicates that the server has attempted proactive content negotiation but does not have enough knowledge to complete the selection, therefore it sends back a list of alternatives to choose from.

This style is advantageous when the response varies over commonly used dimensions (such as media type, language, or encoding), when such preferences are known but not conveyed by the user agent. Also, when caching is used, network overhead is reduced as intermediaries can send back the list of variants without reaching the origin server. In addition, users gain more privacy since no description of preferences and capabilities is sent to the server. However, there are drawbacks to using this pattern. First, latency is increased due to the mandatory round trip to select the final representation. Second, this pattern does not define a common global mechanism to support automatic representation selection, nor available representation listing. In Chapter 8 we partially address this problem to facilitate heterogeneous system interaction in this pattern.

As mentioned above, content negotiation styles are not mutually exclusive. For example, we can combine reactive and proactive styles: A client sends the list of preferences to the server. The server has five different representations and engages in proactive style, but needs more information to choose between two representations. So it sends the shortened list of alternatives to choose from. Finally, the user agent chooses the appropriate representation.

**Request** This is a new content negotiation style introduced as part of the latest HTTP specification (RFC 9110) [61, Section 12.3]. It is called *request* because preferences are sent in a server's response with the intention of influencing subsequent requests for that resource. The `accept-*` headers (see Table 3.1) are examples of means of conveying such intent. For instance, the `accept` header can be sent to indicate the preferred media type for subsequent `POST` requests to that resource.

*HTTP client hints* (or *user-agent client hints*) are a set of HTTP request header fields that a server can use to proactively hint a client about some specific preferences (see the W3C draft community group report on user-agent client hints [172]).

This behaviour is also observed and defined in `PATCH` requests [53, Section 3.1]. The server uses the `accept-patch` response header to indicate to the client the content types it accepts.

HTTP comes with several status codes to allow a server to indicate client errors (via `4xx`), the most generic being `400 Bad Request` to indicate that the server cannot or will not process the request due to some client error, or `403 Forbidden` to indicate that the request has been understood but the server refuses to fulfil it. Two specific status codes can be used to indicate and possibly negotiate acceptable content: `415 Unsuported Media Type` and `422 Unprocessabe Content`. The former, when used with the `accept` response header, indicates which media types would have been accepted in the request. The latter indicates that although the server understood the content type of the request content, it was unable to process the instructions contained within. This is a good candidate for use with the `accept-profile` response header in the profile dimension.[19]

This style is beneficial because it allows a server to express what it considers to be preferred/accepted content. It also minimises bandwidth by avoiding future round trips to negotiate acceptable content. However, as mentioned above, this pattern is newly introduced as part of the HTTP specification (RFC 9110) and is not yet as widely used as proactive and reactive patterns, so sending additional response headers may have no effect on future requests, thus increasing response size in vain.

**Transparent** A request sent from a client to a server typically passes through multiple intermediate parties (e.g. proxy cache). Transparent pattern makes all available alternatives that exist within the origin server visible and gives these intermediaries the ability to negotiate for the best representation on behalf of the client, the user agent, or the origin server [84]. Such selection uses a standardised algorithm

[19] Issue discussing the use of `422 Unprocessabe Content` when negotiating in the profile dimension: https://github.com/ProfileNegotiation/I-D-Profile-Negotiation/issues/40

called *remote variant selection algorithm* [84, Section 7]. Such an algorithm with the version number 1.0. was also defined [83].

A status code introduced by this pattern is `506 Variant Also Negotiates` to indicate that processing the request results in a circular reference, i.e. "the server has the following internal configuration error: the selected variant resource is configured to engage in transparent content negotiation itself, and is therefore not a proper endpoint in the negotiation process." [84, Section 8].

Similarly, several headers have been introduced by the specification (RFC 2295). Among them we find `alternates` to convey the list of variants associated with a negotiable resource. `negotiate`, which contains directives for each content negotiation process initiated by the request. `tcn` used by a server to signal that the resource is being transparently negotiated. And `variant-vary`, which is used in a choice response to record any variable information that applies to the variant data contained in the response, as opposed to `vary` which is used for the response as a whole.

This style has the advantage of reducing response time and bandwidth consumption by distributing the negotiation that would otherwise be required for the whole cycle, i.e. from the client to the origin server and back. Also, the use of caching mechanisms provides a time gain (for more, see the article on the benefits of transport content negotiation in HTTP [155]). However, such an underlined assumption of maximum resource cachability is in practice only true for static and unencrypted content, so it would not be cost effective in contexts where transmissions are encrypted (for more, see the article on the cost of "S" in HTTPS [126]).

**Conditional** Also called *recipient-based* is similar to the reactive one, in the sense that the server sends the available representations and the client, user agent or user makes the final decision. However, whereas in the reactive style only a list of links is sent to choose from and a second request is required to retrieve the actual representation, here the server sends back a multipart representation that contains all the representations, i.e. the body of the response is composed of several parts separated by a boundary. Each one is an alternative representation of the resource. The appropriate parts are then selected/used directly, without the need for additional requests (the HTTP specification briefly mentions this pattern [61, Section 12], and it was previously presented in a document describing scenarios for the delivery of negotiated content [81, Section 4.2]).

Since the body of the response is not homogeneous, we rely on the multipart media types [63, Section 5.1]. Several multipart media types have been registered over the years,[20] two of them are `multipart/mixed` and `multipart/alternative`.

[20]List of registered multipart media types in IANA: https://www.iana.org/assignments/

The former is the primary subtype for multipart, so if any of the subtypes are not recognised, it should be treated as being of subtype `mixed`. The latter is syntactically identical to the former, but semantically different. As its name suggests, each part of the body is an alternative representation of the same information, and the choice of the *best* media type can be based, for example, on the user's environment and preferences.[21]

This style is advantageous for reducing the number of requests to one, in order to obtain a multipart response containing all possible alternatives. However, if the number of variants or the size of the different parts is too large, this pattern becomes less relevant.

**Active** In this pattern, the server includes an executable script in the response. The content, when executed in the receiver's environment, detects the capabilities/preferences of the receiver and makes additional (more specific) manipulations (e.g. additional requests or adaptation procedures) to have the appropriate representation. The HTTP specification briefly mentions this pattern [61, Section 12], and was previously presented in a document describing scenarios for the delivery of negotiated content [81, Section 4.3].

This style has the advantage of reducing the need for user intervention, as additional manipulations are performed automatically. Also, the active content can have complex manipulations and be used to provide and personalise representations to suit the capabilities of the user agent. In addition, such active content can be cached and used in future requests. However, there are some drawbacks to this pattern: First, additional manipulation, e.g. a new request with appropriate `accept-*` headers is required to obtain the final representation. Second, since the scripts are executed on the receiving system, there may be potential threats e.g. stealing the user's credentials, acquiring sensitive data about the user, or attempting to install malware on the user's system. Such a system should prevent the active content from executing arbitrary code. Because of such risks, browsers may prohibit active content by default.

Table 3.2 summarises the content negotiation styles with some references to existing usage examples.

These content negotiation styles are protocol-level patterns and can be applied to

`media-types/media-types.xhtml#multipart`

[21] Note that when using `multipart/alternative` the parts are in ascending order, meaning that the *last* part is the one preferred by the server.

[22] A reference introducing or describing this style, though not necessarily the first which does.

| Style | Reference [22] | Date | Type | Usage |
|---|---|---|---|---|
| Proactive | [15] | 1992 | RFC | [144, 191] |
| Reactive | [15] | 1992 | RFC | [144] |
| Request | [61, Section 12.3.] | 2022 | RFC | [197] |
| Transparent | [84] | 1998 | RFC | [191] |
| Conditional | [81, Section 4.2.] | 1997 | IETF Draft | [118] |
| Active | [81, Section 4.3.] | 1997 | IETF Draft | |

Table 3.2: A summary of the content negotiation styles with some references for usage examples.

any content negotiation dimension. As seen in the previous sections, these patterns allow us to answer questions such as *Who made the choice of representation? (e.g. the server, the user agent, etc.) How should the global response look like? (`300 Multiple Choices` status code for reactive and media type `multipart/mixed` for conditional).* However, they are not sufficient to answer finer questions such as *Does the server use strict or flexible conformance checking?* or *Is validation performed locally by the server or by a third party?* For example, if private data is involved in the negotiation, an answer to such questions would affect the negotiation process. To address this issue (Limitation 3), Chapter 6 introduces and provides guidance on finer patterns that can influence the negotiation process beyond content negotiation styles, with a focus on our dimension of interest, which is profile.

### 3.3.3 Content Negotiation Constraint Conveyance Means

A content negotiation constraint conveyance means denotes the way in which the client communicates its intended negotiation dimensions to the server, as well as preferred values for these dimensions, which should be taken into account in the selecting/generating process of the response to be delivered. Two main techniques have been widely used to perform such transmission: *header-based* and *URI-based*. Note that these constraint conveyance means are not mutually exclusive and can be used together, i.e. a client can send an `accept-language` header and a `format` key-value pair in the URI query part. However, the specifications that allow them to be used together must define the order of precedence. For example, in their document defining content negotiation by profile, the DXWG have chosen URI-based over header-based "This is on the basis that the more precisely a URI points to a resource, the more likely it is that the agent intends to request that precise resource." [161,

Section 6.5]. Next, we give an overview of the various constraint conveyance means with their respective definitions and usage, and point out the advantages and disadvantages associated with using a particular means. The Appendix B.3 provides illustrative examples of different requests using each of the constraint conveyance means.

**Header-based** Headers are essential elements of the HTTP protocol, as they allow additional information to be transmitted by the sender (as request headers) and the receiver (as response headers). Such headers are used to convey preferences in the different content negotiation dimensions (cf. Section 3.3.1). In general, `accept-*` headers are used in requests and `content-*` headers are used in responses, e.g. `accept` and `content-type` for the media type dimension, `accept-language` and `content-language` for the language dimension, or `accept-presentation` along `content-presentation` for the presentation dimension. Note that the header fields usually define a wildcard value to select unspecified values, traditionally denoted by "*". If present, this means that all values are acceptable; if absent, it means that unspecified values are not acceptable [61, Section 12.4.3]. In addition, HTTP allows the sender to weight the preferences of the same dimension by quality values [61, Section 12.4.2]. The example:

```
accept:  image/*; q=0.2, image/png
```

will be interpreted as: I prefer `image/png`, but send me any image type if it is the best available. Usually these values are used together in calculating the overall quality of a representation [84, Section 19.1].

An analysis of the header usage patterns of HTTP requests has been carried out in a previous work [37]. One insight is that `accept` and `accept-language` headers are widely used as a means of negotiating preferred content.

It is worth noting that Constrained Application Protocol (CoAP) also defines content negotiation, but only for formats via the `accept` option [156, Section 5.5.4]. New CoAP options can be defined to accommodate other dimensions.

This constraint conveyance means is advantageous because it allows quality values to be specified for different preferences even for those in the same dimension, thus, adding some flexibility and expressiveness. In addition, headers are generally available in a IANA registry [61, Section 16.3.1], which facilitates interoperability between servers and the automation of the negotiation process. However, it is not straightforward for the average user to express preferences in headers, although this is

sometimes mitigated by the use of plugins.[23] Moreover, headers are not stored in user agents when a web page is bookmarked. Finally, different server implementations have different maximum sizes for header values,[24] When such a limit is exceeded, a server will usually respond with the `413 Content Too Large` status code.

**URI-based** URIs are excellent for providing a simple and extensible way of identifying resources on the Web, but not only that, they could also be used as a way of conveying constraints to guide the selection/generation of a preferred representation. This is achieved in three main manners, namely *path extension*, *Archival Resource Key (ARK)* and *qsa*. We will discuss each of these in the following paragraphs. Note that they are all convenient in the sense that they are easy to edit, even for the average user using the browser's address bar. Also, since bookmarking saves the current URI, it implies that constraints are saved as well. However, there is no standard way to express preference weights as it is the case for headers. They also suffer from the URI's maximum size limit, and if such a limit is exceeded, a `414 URI Too Long` status code is used in the response.

*Path extension*: Also known as *suffix pattern matching*, this approach adds an extension to the URI, primarily to request a specific media type. For example, the URI:

`http://youctagh.com/species/abies_numidica.en.json`

would request the species *Abies numidica* in `application/json` media type and English language. Even though such a method can be generalised to other dimensions, one can only request a single value e.g. one media type and/or one language.

*ARK*: This is an identification scheme for a persistent identifier for information objects. This technique also relies on URI suffixes using the optional *qualifier* part. The specification [99] mentions three types of dimensions (language, version and format) separated by a '.' (dot). For example, the URI:

`http://myarkserver.com/ark:12345/abcd/ef/en.v2.pdf`

would negotiate for the *second version*, *English* and `application/pdf` format of that resource.

[23] Check for example *RDF browser addon* [152] which allows the negotiation and visualisation of RDF representations.

[24] Maximum size of HTTP header values: https://www.tutorialspoint.com/What-is-the-maximum-size-of-HTTP-header-values

| Conveyance Means | Means type | Ref [25] | Date | Type | Usage |
|---|---|---|---|---|---|
| Header-Based | — | [15] | 1992 | RFC | [196] |
| | Path Extension | [189] | 1996 | Blog Article | [196] |
| URI-Based | ARK | [100] | 2005 | IETF Draft | [178] |
| | QSA | [18] | 1998 | RFC | [196] |

Table 3.3: A summary of the content negotiation constraint conveyance means with some references for usage examples.

*Query String Argument (QSA)* A query string in a URI takes the form of "key=value" pairs, commonly used to provide additional information to a server. One type of such information is negotiation preferences. For example, the URI:

```
http://example.com/resource/?format=text/turtle&language=en
```

would negotiate for the `text/turtle` format with the *English* language. Note, however, that these keys are for illustrative purposes only, and there are no universal keys defined by the specification [20].

Table 3.3 summarises the content negotiation constraint conveyance means with some references to existing usage examples.

### 3.3.4 Content Negotiation in Practice and Existing Contribution Benchmark

The benefits and necessity of leveraging content negotiation are highlighted in several W3C documents, for instance by the Spatial Data on the Web Working Group [34], and by the Web Data Best Practices Working Group [55] in their best practices documents. Many server implementations offer the ability to perform content negotiation. One of the earliest is the Apache server [189]. Content negotiation was possible in three dimensions: media type, language, and encoding using file variants or file extensions. Frameworks such as Spring [196] also provide the ability to configure and execute content negotiation through the `accept` header, path extensions, or even by configuring a preferred parameter to be used in a URI-based content negotiation. Furthermore, Spring has the ability to extend these implementations by

[25] A reference introducing or describing this constraint conveyance means, though not necessarily the first which does.

| Ref | Date | Style | Dimension | Means | Protocol |
|---|---|---|---|---|---|
| [36] | 2001 | Proactive | Capacity | Header | HTTP |
| [177] | 2004 | Proactive | Language | QSA, Headers | HTTP |
| [178] | 2018 | Reactive, Proactive | Media type, profile | ARK | HTTP |
| [91] | 2018 | Reactive, Proactive | Time | Header | HTTP |
| [104] | 2018 | Proactive | Presentation | Header | HTTP, CoAP |
| [118] | 2019 | Conditional | Media type | Header | HTTP |

Table 3.4: Some contributions using content negotiation and their characteristics.

intercepting custom headers and URI parameters to create a custom content negotiation process. Most frameworks, including *ASP.NET* [193], *Django* [194], *Play* [195] also implement content negotiation.

We can also find some practical uses of content negotiation in the Semantic Web domain: RDF Browser [152] is a Firefox add-on that adds appropriate headers to negotiate RDF content. Vapour [24] is a validation service for verifying that Semantic Web data is correctly published according to common best practices. The SPARQL 1.1 protocol [56, Section 2.1] states that the response to a query can be either the SPARQL (XML, JSON or CSV/TSV) result formats or an RDF serialisation, depending on the query form and content negotiation.

In the body of literature there are several contributions that use content negotiation as part of their process. The content negotiation characteristics presented in the previous sections can be used collectively to classify them and provide a global view of the nature of the interaction between the parties involved (e.g. user agent, server). Table 3.4 shows examples of such classification. A row can be read as follows: the contribution with the reference {`ref`} published in year {`year`} allows negotiation in dimensions {`dimensions`}, using styles {`styles`}, where constraints were conveyed using means {`means`}. A final column indicates the protocol used, since HTTP is not the only web protocol.

CC/PP and UAProf are descriptions of device capabilities and user preferences. [36] describes an implementation of content negotiation in HTTP that uses these descriptions to provide the best variant. The dimension of negotiation is capacity, the headers are used to convey constraints, and the proactive style is used.

Google Scholar [177] is a well-known search engine that indexes scholarly literature. Page content can be proactively negotiated in multiple languages using the HTTP `accept-language` header or the `hl` query parameter with precedence for the

QSA over headers.

Istex is a French scientific archive [178]. The clients have the possibility to request the representation in several media types and profiles. To achieve this, ARK is used over HTTP. If we request the resource without specifying the media type, we get a JSON file describing the available variants and therefore we can consider it a reactive and proactive negotiation.

The goal of the paper [91] by Kelly et al. is to allow a client of a memento aggregator to specify a custom set of archives to aggregate. To achieve this, the process goes beyond the traditional way of specifying just a date, and involves specifying a set of archives in the `prefer` header. So the characteristics are time for the dimension, headers for the constraint transportation, and HTTP as the protocol. Since the aggregator returns a set of records and the client can potentially manipulate the response to make another request, we consider this part to have a reactive and proactive content negotiation style.

Lefrançois [104] describes a scenario where a client requests a resource representation from a server and wants the response to be encoded according to a specific RDF presentation, which is done by including metadata in the `accept-presentation` request header. And since this article is primarily intended for constrained devices, the proactive style is preferable. The author points out that although this method is implemented in HTTP, it could easily be defined as equivalent CoAP options.

Using the MarkLogic [118] server, a REST application developer uses the conditional content negotiation style. A client would receive a response with a body containing several parts separated by a delimiter to be specified. This method is primarily used to select the media type dimension and uses the HTTP `accept` header to communicate this preference.

## 3.4 A Closer Look on the Profile Dimension

Before presenting our proposal to enable relaxed and flexible semantic content negotiation, we need to better understand the evolution of negotiating semantic content on the Web. To do this, we focus on the profile dimension, since it is the one concerned with this type of preference. We review existing contributions related to content negotiation by profile, with a particular focus on the ongoing efforts led by the DXWG [161] and the IETF [163].

The DXWG describes several use cases that require profile-based content negotiation [140, see for example use cases `ID2` and `ID3`]. However, for the Semantic Web community, the use case of requesting an RDF representation with some expected vocabularies is older. In 2006 already, a thread on the Semantic Web mailing list

discussed the topic.[26] Two essential components are important to convey the preferences of the client: a way of representing the preferences (in our case, the definition of a profile) and a way of transmitting them. We examine the state of the art for the two components.

### 3.4.1 Specifying a Profile

HTTP currently defines content negotiation in four dimensions: media type, language, encoding, and character set [61, Section 12.5]. Among these, media types can serve as a way to specify a profile. Media type definitions can add certain structural constraints to a more generic format, hence creating a "profile" for it. For instance, in its XML Media Types proposal [125], the IETF network working group introduced the use of the convention of adding (`+xml`) for newly created media types that are XML-based, which allows for generic XML processing.[27] This is a case of adding semantics to the generic XML document. This suffix convention has since become the de facto practice for other structuring syntax (e.g. `+json`, `+zip`, `+jwt`) [64, Section 4.2.8]. Even if we ignore the constraints of the required registration process, it is still impractical to define a new media type for all possible profiles, especially with a broad sense of "profile" where constraints could be customised per application. Moreover, a profile would then be tied to the media type, whereas the two dimensions should be orthogonal, which is why we can observe that some media types have the same prefix but differ only in their suffixes (e.g. `senml+xml`, `senml+exi`, `senml+json`, `senml+cbor`). In this case, if a client wants to negotiate a profile disregarding its serialisation, it must specify all possible media types.

In principle, it could be possible to append vocabulary prefixes to the media-type token (e.g., `text/foaf+schema+org+turtle`) to form more fine-grained constraints while keeping the exact same architecture for content negotiation. However, the limitation of such approach should be obvious and still this would not cover many more kinds of specific constraints that one may have, as highlighted by Verborgh [175]. This is also known as the media type explosion problem.[28]

Nevertheless, the difficulty of expressing finite constraints to create a profile in the Semantic Web is encountered even in the simplest cases [175]. For example, representations of people that conform to a profile that use only three different

[26] Semantic Web mailing list archives: *expectations of vocabulary* https://lists.w3.org/Archives/Public/semantic-web/2006Jul/0062.html

[27] At the time of writing, there are more than 400 registered media types using this convention for XML documents.

[28] Media type explosion problem (last paragraph): https://lists.w3.org/Archives/Public/semantic-web/2006Jul/0097.html

vocabularies (e.g. `FOAF` [33], `Schema.org` [192], `Organisation` [143]) and only one serialisation (e.g. `Turtle`). How should media types be defined?

`text/v_foaf_person+v_schemaorg_place+v_org_organisation+turtle`

or `text/v_org_organisation+v_foaf_person+v_schemaorg_place+turtle`

and what would be the relationship of say `text/v_foaf_person+turtle` with them. From a syntactic point of view, it can be said that there is a containment relationship between the one and the other. However, this can get out of hand very quickly.

Indicating profiles in the media type is one way of specifying a profile. The W3C CC/PP [94] is an RDF-based standard for describing device capabilities and user preferences. A server can use such information to customise the served content. FOAF has been proposed as a way to define a personal profile that can be exploited by servers to improve Web browsing experience [6]. The DXWG defines a vocabulary that can express metadata about profiles [8]. It also mentioned that a constraint or schema language (e.g. SHACL [97]) can be used to specify a profile [163]. The group also notes that a human-readable document (e.g. in HTML or PDF) may serve as the specification of a profile. In each of these cases, the profile can be referred to by serialising it appropriately in a payload, or by way of a URI.

While it is preferable to identify profiles by URIs that dereference to a description [161, Section 6.2][163, Section 4.1], it is possible to use URIs that do not. However, means should be provided for clients and servers to understand their meaning. A server can advertise a repository containing the profiles it can handle, e.g. SpacePrez[29] or CC/PP repositories [94, Section A.1].

Some profiles may define a set of constraints of varying importance. In a content negotiation implementation using CC/PP and UAProf [36], the author extends the Apache web server by providing the ability to configure the constraints as a *hard* or *soft* constraint using a boolean `mustmatch` attribute. Similarly, in SHACL, the specification provides the means to define a *severity* for each shape [97, Section 2.1.4]. This offers ways of enabling more flexible negotiation, as we will see in a later chapter, (cf. Section 6.1.4).

Going back to the example of using a header-based approach to convey a profile, specifically the example: "`accept-vocabulary`: `http://xmlns.com/foaf/0.1/`", one can think of various semantics for such a header, one of which is: I only accept statements that use FOAF predicates. However, the constraints could be finer (e.g. I want instances of the class `foaf:Person` to have at most one value for the property `foaf:name`). In that case, constraint languages might be used, as they allow the definition of constraints to which the representation must conform. Examples of such constraint languages are: SHACL [97], ShEx [137], JSON Schema [186], XML

[29]Profiles recognised by SpacePrez: `https://gnaf.linked.fsdf.org.au/profiles`

Schema [66]. A URI can then be used to point to that constraint document.[30]

In a content negotiation process, if the client's constraint cannot be satisfied, the server may send a `406 Not Acceptable` status code [61, Section 15.5.7] to indicate that no acceptable representation could be found. However, the two parties can minimise the occurrence of such conclusion. The client can give some fallback options, for example in HTTP by negotiating a group of media types such as `image/*` instead of specifying all image media types (`image/png`, `image/jpeg`, etc.). A wild card `(*)` could be used in the language dimension [134]. Using the profile vocabulary proposed by DXWG, a profile hierarchy could be constructed through the use of `isProfileOf` property giving the possibility to use the upper profiles as fallback options if the initial profile does not conform [8, Section 8.3.2]. The server on the other hand may adapt a representation to fit the request [110, 111].

### 3.4.2 Conveying Profile Preferences

A client expecting conformance to profiles must convey them to the server. We build on top of the DXWG group to classify existing approaches [161, Section 5]. Two main options are found in the state of the art: Header-based;[31] and URI-based.

#### Header-based.

The DXWG identified several existing standards that rely on HTTP headers and that could be used to convey profile preferences, but argued that they are insufficient for fully capturing the needs of content negotiation by profile. Media types can provide additional parameters [61, Section 12.5.1] that can be appended to the value of the `accept` header, particularly the `profile` parameter (e.g. `application/ld+json ;profile="http://www.w3.org/ns/json-ld#flattened"`).[32] However, not all media types allow such parameter and a profile of this kind is media-type-specific.

The `link` header [129] together with the `rel="profile"` relation type [182], as well as the `prefer` header [157] could provide a profile preference, but neither can convey quality information (q-values) [61, Section 12.4.2]. Note, that the `prefer` header also allows for processing preferences that have hard and soft constraints using

[30] Since a profile can play different roles, not just validation and constraint definition, the DXWG proposes to assign roles to profile resources and provide an extensible list of roles [8, Section 9].

[31] Note that this header-based approach is aligned with the suggestion in the XML media type definition [125, Section A.10] "How about using a content negotiation tag instead (e.g. `accept-features`: syntax=xml)?".

[32] RFC 7284 [102] defines a IANA registry for profile URIs https://www.iana.org/assignments/profile-uris/profile-uris.xhtml

`handling=strict` and `handling=lenient` [157, Section 4.4]. The `accept-profile` header [175] is now DXWG's preferred proposal [161, Section 7.2] for conveying preferences regarding resource representation profiles. A separate IETF document explains in more detail how to indicate, discover, negotiate and write profiled representations [163]. If multiple profiles are sent in a request, q-values can be used to convey the preference for each profile. However, note that these preferences are applied at the profile level. For example, if we consider a SHACL profile composed of multiple shapes, q-values cannot be used to give different weights to individual shapes Limitation 1.

Other headers that we have already discussed were suggested (see Table 3.1) e.g. `profile`, `accept-schema` and `accept-vocabulary`.

`accepts-vocab` [72] is similar to the last one, but for JSON representations instead of RDF ones. Similarly, `xfoaf` [6] was a header proposal used to allow a client to communicate its FOAF profile to a server.

### URI based.

*Query string arguments* can be used to convey profile preferences in the URI [20, Section 3.4], e.g. `http://domain.com/path/to/res?profile=profile_info`. Here, when negotiating for the resource identified by `http://domain.com/path/to/res`, profile preference is conveyed using query parameter `profile`. Such use can be observed in several settings.[33,34,35] The DXWG proposes the `_profile` query parameter to convey profile preferences [161, Section 7.3].

*Suffix pattern matching* uses the part of the URI after a specific character to convey profile preferences, e.g. `http://domain.com/path/to/res.profile_info`. In this example we use '`.`' as a special character to indicate the start of the preferred profile. Other conventions could be used to signal the end of the resource identification and the start of the profile preferences.

*ARK identifier scheme* permit the uses of the qualifier part to specify the preferred profile. Istex uses such an approach to serve various XML representations: plain XML, text encoding and interchange [146] and the metadata object description schema [75].

[33] Linked Data API using the `_view` query parameter in the negotiation process: `https://github.com/UKGovLD/linked-data-api/blob/wiki/Specification.md`

[34] The Open Archives Initiative Protocol for Metadata Harvesting uses the `verb` query parameter [101].

[35] The open geospatial consortium catalogue service for the web standard uses the `outputSchema` query parameter [127].

Examples of some of the aforementioned approaches have been provided within the DXWG document as well as the IETF document [163]. Note that for DXWG, tokens could be used instead of URIs to identify profiles [161, Section 6.2], where a token is a short textual identifier. However, in the IETF specification [163, Section 4.1], profiles must be identified by a URI.

Although these different contributions differ in their final objectives, the common point that unites them is the use of content negotiation, as highlighted over the years by several working groups in their best practice documents [23, 55, 34], where each use case involves some content negotiation finer patterns. However, to the best of our knowledge, no previous work has attempted to compile the different content negotiation patterns. In particular, to illustrate how they can be used in the profile dimension, and how semantic validation can be applied and integrated in different content negotiation scenarios. This is what we will present and discuss in Chapter 6.

## 3.5 Synthesis

In this chapter we have presented a survey on the practice of content negotiation. We discussed several aspects that are relevant to better identify the limitations that motivate our work (cf. Section 1.1).

In Section 3.1, we presented the different stages of content negotiation, emphasising that it is not a monolithic process, but rather a multi-layered one, and that these stages should guide us to achieve the envisioned flexible and relaxed content negotiation.

In Section 3.2, we examined attempts to formalise content negotiation and concluded that there is a need for a generic model that should be extendable to the Semantic Web to achieve semantic content negotiation using semantic validation languages.

In Section 3.3 we surveyed the characteristics of content negotiation. We introduced the different dimensions of content negotiation over which representations can vary, the styles by which the content negotiation process is carried out, and possible ways of conveying constraints to be considered in the negotiation process. We have seen that while these characteristics have allowed us to answer global questions about the content negotiation process (e.g. at the protocol level), they are not sufficient to answer finer questions (e.g. at the validation level). We then looked at examples of some practical applications and implementations of content negotiation.

In Section 3.4 we focused on the profile dimension, which allows for the negotiation of semantics. We based our survey on the recent efforts of the DXWG and IETF to present the known ways of specifying and transmitting a profile. We found that

previous existing work tried to relax the negotiation by defining two types of capacity constraints (using CC/PP and UAProf): hard and soft. However, such an idea is non-existent when checking representation conformance using semantic validation, despite the fact that a language such as SHACL defines the concept of severities that could be used to accommodate such flexibility.

We conducted a survey on content negotiation [164, 170], and as a result we created the Content Negotiation Theoretical Framework (CNTF) resource [167]. CNTF is a website that collects knowledge about content negotiation use cases, styles, dimensions, etc. and organises it according to an ontology. CNTF aims to highlight existing solutions, where available, or suggest plausible ways to meet common requirements. It is also intended to be used to disseminate future proposals for the development of content negotiation, making it a sustainable and up-to-date digital survey of content negotiation. Appendix A provides more details about this resource.

In Part II, we discuss our contributions towards the design of a flexible and relaxed content negotiation.

# Part II

# Contribution - Design a Flexible and Relaxed Content Negotiation

Chapter 4

# CON-NEG – A Model for Content Negotiation

## Contents



In Part I of this dissertation, we analysed the state of the art with the objective of understanding the beginning and the evolution of content negotiation on the Web. We discussed several limitations that hinder the development of an extended semantic-based version, which should be flexible, relaxed and guided by the content negotiation stages (cf. Section 3.1).

In this part we focus on the modelling and design aspect of our semantic content negotiation framework. In this chapter we propose an abstract model for content negotiation we call CON-NEG (for CONtent NEGotiation). In Section 4.1, we start by presenting the first part of an incremental content negotiation use case, which allows us to frame what falls within the scope of our proposal. In Section 4.2, we provide our formalisation of the content negotiation problem, which captures common practices of content negotiation on the Web in the classical dimensions present in the HTTP protocol. We then show how it could be extended in the

context of the Semantic Web to allow the negotiation of RDF graphs that conform to some profiles. Finally, in Section 4.3, we discuss the applicability of CON-NEG in two contexts: first in HTTP with an apache server, the second applied with an Object-Oriented Programming (OOP) paradigm.

In this chapter, we use relative IRIs that resolve against the base IRI: `http://youctagh.com/`

## 4.1 Incremental Use Case - Part I

On the Web, a general case of content negotiation happens when a client requests a representation of a resource (by issuing an HTTP `GET` request) accompanied by constraints that the returned representation should satisfy. If a server has multiple representations for the requested resource, a negotiation phase occurs: in some cases, the server decides which representation is the most appropriate given the constraints and returns it directly; sometimes the server provides a redirect to another resource or representation that might satisfy the client; sometimes the server simply informs the client of all available representations, possibly with weights indicating its preferences. In the current state of the formalisation, we focus on the first case, i.e. the proactive style presented in Section 3.3.2.

To better frame the considered scenarios, we present an incremental use case. It builds on the stories presented in Part I. In Section 4.1.1 we consider the case of negotiation in the classical dimensions: media type and language. In Section 4.1.2 we cover the case of negotiating RDF representations by profile.

### 4.1.1 Content Negotiation in the Classical Dimensions

*YoucTagh* provides six different *representations*[1] of the *resource Abies numidica identified* by the IRI `<species/abies_numidica>`: (1) HTML; (2) PDF; (3) a simple text in French; (4) a simple text in English; (5) RDF serialised in Turtle; (6) same RDF but serialised in RDF/XML.[2]

*YoucTagh* hosts various resources on his *web server* that accepts request queries from web clients. A *query* to negotiate one of the available resource representations

[1]Terms are emphasised to highlight the different concepts that will be present in the abstract model.

[2]The IRI template `<species/abies_numidica.{language_tag}.{media_type_extension}>` is used to identify the different representations, e.g. `<species/abies_numidica.en.txt>`. If the language is not specified, the `.language_tag` part is omitted, e.g. `<species/abies_numidica.rdf>`.

is accompanied by some *constraints* to negotiate for the appropriate representation in some particular dimensions. *YoucTagh* also expresses the quality of the available representations in the form of internal *constraints*.[3]

A client can express for example a preference for:

1. an RDF/XML representation;
2. any RDF representation;
3. an Image representation. If not available then a Textual representation;
4. a French representation;
5. an Arabic representation. If not available then an English representation;
6. a Turtle representation. If not available, then any Textual representation;
7. a Textual representation. However, not a Turtle one.

Having both the client's constraints and its own internal constraints, *YoucTagh* uses a *measure* (a.k.a *CN measure*) to calculate the final qualities of the representations, and subsequently selects the best one.

### 4.1.2 Content Negotiation in a Semantic Context

*YoucTagh*, as the creator of the above representations, wants to provide additional information about himself and link to it when necessary. He decides to use three different vocabularies: `FOAF`, `Organisation` and `Schema.org`. He gives the ability to negotiate one of three different RDF representations conforming to the following profiles: (1) uses only `FOAF`; (2) uses `FOAF` in addition to `Organisation`; (3) uses only `Schema.org`. The profiles are also identified by IRIs.

For example, a client requesting a representation that conforms to the FOAF profile is guaranteed to receive a representation that: (1) has at least one `foaf:Person` instance; (2) all instances of `foaf:Person` contain one `foaf:familyName` and one `foaf:givenName` predicate. Listing 4.1 is a snippet of the FOAF profile expressed in SHACL.

[3]The quality can be thought of as the server's preferences and prior knowledge. For example, if the PDF representation does not capture as much knowledge as the HTML representation, it would have a lower quality. The same applies if the translation is not as accurate.

Listing 4.1: A snippet of the FOAF profile expressed with SHACL

```
ytshape:FoafPersonMinCount a sh:NodeShape;
    sh:targetNode foaf:Person;
    sh:property [
        sh:path [sh:inversePath rdf:type ;];    sh:minCount 1;
    ].
ytshape:PersonShape  a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:familyName ;  sh:minCount 1 ;
    ];
    sh:property [
        sh:path foaf:givenName ;  sh:minCount 1 ;
    ].
```

The RDF representations and their corresponding profiles are as follows:

1. The `FOAF` representation: `<me/yt-foaf.ttl>` conforms to the profile `<profiles/prof-foaf.ttl>`
2. The `FOAF` + `Org` representation: `<me/yt-foaf-org.ttl>` conforms to the profile `<profiles/prof-foaf-org.ttl>`
3. The `Schema.org` representation: `<me/yt-schema.ttl>` conforms to the profile `<profiles/prof-schema.ttl>`

## 4.2 A Formalisation of the Content Negotiation Problem

In this section, we want to formalise the content negotiation problem in a way that encompasses current practices and implementations, but also allows for more fine-grained negotiation. So far, HTTP clients and servers have mostly implemented negotiation by media type, language, encoding and charset (those defined by the HTTP specification). In what follows, we first abstract away from these specific dimensions of negotiation in order to generalise them. Then we study the case of negotiation of RDF representations conforming to some profiles (e.g. expressed in SHACL). We divide this section into two parts: First, we formalise the problem of generic content negotiation, transforming the components of the content negotiation mechanism into mathematical structures and functions (Section 4.2.1). Second, we propose an extension of the abstract model to RDF graphs and profiles (Section 4.2.2).

### 4.2.1 CON-NEG – A General Abstract Model

In what follows, we structure our discussion in a bottom-up manner: we first define each element present in the negotiation process, and then introduce a formal definition of the content negotiation problem. Throughout the rest of this section, and for illustration purposes, we use the scenario described in Section 4.1.1.

**Definition 4.1: Resource**

We define the set of all *resources* $\mathcal{R}$. A resource $r \in \mathcal{R}$ can be a person, an organisation or anything identifiable.

For example, in our scenario, *YoucTagh* and *Abies numidica* are resources.

**Definition 4.2: Identifier**

We define the set of all *identifiers* $\mathcal{I}$. An identifier $i \in \mathcal{I}$ is used to identify a resource. The identification function $id$ is formalised as follows:

$$id : \mathcal{I} \rightarrow \mathcal{R}$$

For example, in our scenario, the IRI $i_{an} =$ `<species/abies_numidica>` identifies the resource *Abies numidica*.

**Definition 4.3: Representation**

We define the set of all *representations* $\mathcal{D}$. A representation $d \in \mathcal{D}$ captures the state of a resource and can be identified. Note that a resource can have more than one representation.

For example, in our scenario, the resource *Abies numidica* has a set of six representations $\mathbf{D}_{\text{an}} \subset \mathcal{D}$ each with its own unique identifier:

1. HTML: `<species/abies_numidica.html>`
2. PDF: `<species/abies_numidica.pdf>`
3. French plain text: `<species/abies_numidica.fr.txt>`
4. English plain text: `<species/abies_numidica.en.txt>`

5. RDF in RDF/XML: <species/abies_numidica.rdf>

6. RDF in Turtle: <species/abies_numidica.ttl>

A naive approach to define the set of constraints is $\mathcal{C} = 2^{\mathcal{D}}$, meaning that a constraint $c \in \mathcal{C}$ defines a set of representations that satisfy that constraint. Consequently, a representation $d \in c$ means that the representations $d$ satisfies the constraint $c$.

However, in practice we need to order the constraints, as in the following request: *(5) I prefer an Arabic representation. If none is available, give me an English one.* To satisfy this ordering requirement, we use the following definition instead:

**Definition 4.4: Constraint**

$\mathcal{C}$ is the set of *constraints*. A constraint $c \in \mathcal{C}$ maps a representation to an acceptability value in the range $[0, 1]$,[a] formally:

$$c : \mathcal{D} \to [0, 1]$$

[a] Just as suggested in the HTTP specification, we can use three decimal places. 0 is unacceptable, 0.001 is the least acceptable and 1 is the most acceptable [61, Section 12.4.2]

Since in practice a client does not know the set of all representations available in a server, a more practical approach is to apply an acceptability value to a subset of representations (e.g. representations with media type `image/png`, or representations using the English language).

Given a set $\mathbf{D}$ of representations, $c_{\top}^{\mathbf{D}}$ and $c_{\emptyset}^{\mathbf{D}}$ are two special constraints. The former assigns a quality value of 1 to all representations in the representation set $\mathbf{D}$. When expressed by a client, this means that the representations are the most desirable. On the other hand, when expressed by a server, it means that they are the best available representations. The latter associate the quality value 0 for all representations in the representation set $\mathbf{D}$. If expressed by a client, this means that the representations are undesirable. If expressed by a server, it means that these representations are unavailable.

These special constraints can be used to define the following four client semantics:

I accept everything:

$$c_{\top} : d \mapsto 1 \qquad \text{for all } \; d \in \mathcal{D}$$

I accept nothing:

$$c_{\emptyset} : d \mapsto 0 \qquad \text{for all } \; d \in \mathcal{D}$$

I accept a set of representations $\mathbf{D}$:

$$c_{\top}^{\mathbf{D}} : \begin{cases} d \mapsto 1 & \text{if } d \in \mathbf{D} \\ d \mapsto 0 & \text{if } d \in \mathcal{D} \setminus \mathbf{D} \end{cases}$$

I reject (or accept except) a set of representations $\mathbf{D}$:

$$c_{\emptyset}^{\mathbf{D}} : \begin{cases} d \mapsto 0 & \text{if } d \in \mathbf{D} \\ d \mapsto 1 & \text{if } d \in \mathcal{D} \setminus \mathbf{D} \end{cases}$$

In our scenario, the constraints set in the request query (5) from Section 4.1.1 we note $\mathbf{C}_5$ can be expressed as follows:

$$\mathbf{C}_5 = \{c_e, c_a, c_{\emptyset}\}$$

Given $\mathbf{D}_e$ the set of all English representations, $\mathbf{D}_a$ the set of all Arabic representations, the two constraints $c_e$ and $c_a$ are defined as follows:

$$c_e(d) = \begin{cases} 0.5 & \text{if } d \in \mathbf{D}_e \\ 0 & \text{otherwise} \end{cases} \qquad c_a(d) = \begin{cases} 1 & \text{if } d \in \mathbf{D}_a \\ 0 & \text{otherwise} \end{cases}$$

Alternatively, we can define a single constraint $c \in \mathcal{C}$ that handle all the above cases as follows:

$$c(d) = \begin{cases} 0.5 & \text{if } d \in \mathbf{D}_e \\ 1 & \text{if } d \in \mathbf{D}_a \\ 0 & \text{otherwise} \end{cases}$$

**Definition 4.5: Request**

We define the set of all *requests* $\mathcal{Q}$. A request $q \in \mathcal{Q}$ is a pair of an identifier with a set of constraints,[a] formally:

$$\mathcal{Q} = \mathcal{I} \times 2^{\mathcal{C}}$$

[a] If no constraint is specified in a request (i.e. the constraint set is the $\emptyset$), this is translated as "I accept any representation".

In our scenario, the client who wants to request a representation of the resource *Abies numidica* and has the language preferences (5) specified earlier, can use the request $q_5$ defined as follows:

$$q_5 = \langle i_{an}, \mathbf{C}_5 \rangle$$

**Definition 4.6: CN Measure**

We define the set of all *CN measures* $\mathcal{M}$. A CN measure $m \in \mathcal{M}$ is a function that associates the final acceptability value to a representation given two sets of constraints. The first is obtained from the client's request and the second is the server's set of constraints. Formally:

$$m : \mathcal{D} \times 2^{\mathcal{C}} \times 2^{\mathcal{C}} \rightarrow [0, 1]$$

For example, in our scenario, suppose the server uses a constraint set $\mathbf{C}_s$ that associates the following acceptability values to the available representations $\mathbf{D}_{\text{an}}$:

1. <species/abies_numidica.html>: 1
2. <species/abies_numidica.pdf>: 0.8
3. <species/abies_numidica.fr.txt>: 0.6
4. <species/abies_numidica.en.txt>: 0.7
5. <species/abies_numidica.rdf>: 0.75
6. <species/abies_numidica.ttl>: 0.85
7. and 0 otherwise.

and let us consider the CN measure $m_{\text{pmax}}$, that calculates the final acceptability value by multiplying the two maximum values obtained when applying a representation to the client's set of constraints $\mathbf{C}_{\text{client}}$ and the server's set of constraints $\mathbf{C}_{\text{server}}$. Formally:

$$\forall d \in \mathcal{D}, \quad m_{\text{pmax}}(d, \mathbf{C}_{\text{client}}, \mathbf{C}_{\text{server}}) = \max_{c \in \mathbf{C}_{\text{client}}} c(d) \times \max_{c \in \mathbf{C}_{\text{server}}} c(d)$$

If we consider the constraint sets $\mathbf{C}_s$ and $\mathbf{C}_5$ (which specify language preferences, see previous definitions) and the CN measure $m_{\text{pmax}}$, we will obtain the final acceptability values:

1. <species/abies_numidica.html>: $1 \times 0 = 0$
2. <species/abies_numidica.pdf>: $0.8 \times 0 = 0$

3. <species/abies_numidica.fr.txt>: $0.6 \times 0 = 0$
4. <species/abies_numidica.en.txt>: $0.7 \times 0.5 = 0.35$
5. <species/abies_numidica.rdf>: $0.75 \times 0 = 0$
6. <species/abies_numidica.ttl>: $0.85 \times 0 = 0$

**Definition 4.7: Server Data**

We define the set of all *server data* $\mathcal{S}$. A server data $s \in \mathcal{S}$ is a triple consisting of a set of resource representations, a CN measure to compute the final acceptability values for those representations, and a set of constraints of that server. Formally:

$$\mathcal{S} = 2^{\mathcal{D}} \times \mathcal{M} \times 2^{\mathcal{C}}$$

For example, in our scenario, the server data of the *Abies numidica* resource is the triple $s_{\text{an}}$ consisting of the set of six representations $\mathbf{D}_{\text{an}}$, the CN measure $m_{\text{pmax}}$, and the set of constraints $\mathbf{C}_s$:

$$s_{\text{an}} = \langle \mathbf{D}_{\text{an}}, m_{\text{pmax}}, \mathbf{C}_s \rangle$$

**Definition 4.8: Web Server**

A *web server* $w$ is a partial function over the set of identifiers $\mathcal{I}$, which assigns to some identifiers the server data of the resource identified by those identifiers, formally:

$$w : \mathcal{I} \nrightarrow \mathcal{S}$$

The set $\mathcal{W}$ of all web servers corresponds to our model of the Web. Note that only one web server can serve any given identifier. Therefore the domains of the web servers are pairwise disjoint:

$$\forall i \in \mathcal{I}, \forall w_1, w_2 \in \mathcal{W}, i \in \text{dom}(w_1) \land i \in \text{dom}(w_2) \implies w_1 = w_2$$

Following this remark, we define the partial function *web*, which is the union of all web servers, formally:

$$web = \bigcup_{w \in \mathcal{W}} w$$

For example, in our scenario, *YoucTagh*'s web server $w_{\text{an}}$ handles the IRI of the *Abies numidica* resource and maps $i_{an}$ to the server data $s_{\text{an}}$ (as defined in the previous definitions). Formally:

$$w_{\text{an}} : i_{an} \mapsto s_{\text{an}}$$

**Definition 4.9: Content Negotiation Problem**

A *content negotiation problem* can be defined as follows. Given:

- A web server $w \in \mathcal{W}$ on the Web serving resources identified by identifiers from the set $\mathbf{I}_w \subset \mathcal{I}$, such that for all $i \in \mathbf{I}_w$: $w(i) \coloneqq \langle \mathbf{D}_i, m, \mathbf{C}_s \rangle$
- A request $q \in \mathcal{Q}$ for a resource identified by $i_q \in \mathbf{I}_w$, with a set of constraints $\mathbf{C}_c \subseteq \mathcal{C}$. i.e. $q \coloneqq \langle i_q, \mathbf{C}_c \rangle$

The content negotiation problem aims to find a representation $d_{\text{best}}$ such that:

$$d_{\text{best}} = \underset{d \in \mathbf{D}_{i_q}}{\text{argmax}}(m(d, \mathbf{C}_c, \mathbf{C}_s))$$

Informally, we define a content negotiation problem as the process of selecting a representation $d_{\text{best}}$ of a resource from a set of representations available at a web server $w$, in such a way that the selected representation maximises $w$'s CN measure $m$ when applying the constraint sets (client preference) $\mathbf{C}_c$ and (server preference) $\mathbf{C}_s$.

In our scenario, given the request $q_5$ and the web server $w_{\text{an}}$, at the end of the content negotiation process, it is the representation *English plain text* that is selected.

In practice, $d_{\text{best}}$ does not always exist. For example, if $\mathbf{D}_i = \emptyset$. We define $d_{\text{no\_rep}}$ to indicate that no representation could be found for the given request.

Note that a server using a proactive content negotiation style will typically respond with a representation of the requested resource, and may provide additional information based on certain parameters that were taken into account in making the selection. However, these parameters are difficult to enumerate and specify, and have been left outside the scope of this general formalisation. Nevertheless, in Section 5.3 we propose a complement in the specific context of content negotiation by profile.

Note also that this definition assumes that the CN measure of a server $m$ is the same for all the resources it serves. In Section 4.3 where we discuss the applicability of our model, we see that this reflects how some servers handle content negotiation.

### 4.2.2 General Model Extension to Semantic Context

In this section we consider the scenario described in Section 4.1.2. First, we illustrate how our abstract model can be instantiated in the Semantic Web context. Then, in order to accommodate some specific cases, we show how the model can be extended. Again, we structure our discussion in a bottom-up manner, introducing new formal definitions as needed.

In this scenario, the *resource* of interest is *YoucTagh*, which is *identified* by the IRI $i_{yt}$ = `<me/>`.

Here we have three different *representations*, also identified by IRIs (see the scenario description in Section 4.1.2):

1. `<me/yt-foaf.ttl>`
2. `<me/yt-foaf-org.ttl>`
3. `<me/yt-schema.ttl>`

However, in this case, a representation is an RDF graph that conforms to a profile. Next, we introduce the concept of profile.

**Definition 4.10: Profile**

We define the set of all *profiles* $\mathcal{P}$. A profile $p \in \mathcal{P}$ describes the structural and semantic constraints of a class of representations and can be identified. If a representation respects the constraints imposed by a profile, we say that it *conforms to* it.

For example, in our scenario, the profile identified by the following IRI: `<profiles/prof-schema.ttl>` describes the structural and semantic constraints of the representation identified by the IRI `<me/yt-schema.ttl>`.

**Definition 4.11: RDF Representation**

We define the set of all *RDF representations* $\mathcal{D}_g$ as the subset of all representations that serialise an RDF graph. Formally:

$$\mathcal{D}_g \subseteq \mathcal{D}$$

In addition we define the constraint $c_{\text{rdf}} \in \mathcal{C}$ that each RDF representation

$g \in \mathcal{D}_g$ must satisfy:

$$c_{\text{rdf}}(d) = \begin{cases} 1 & \text{if } d \in \mathcal{D}_g \\ 0 & \text{otherwise} \end{cases}$$

In our scenario, for simplicity, an RDF representation conforms to only one profile. However, in practice it may conform to several.

Assume the client wants to request the FOAF representation. First, we define the constraint set $\mathbf{C}$, which contains the $c_{\text{rdf}}$ constraint to filter out any non RDF representations, as well as the constraint $c$ defined as follows:

$$c(g) = \begin{cases} 1 & \text{if } g \in \mathbf{G} \\ 0 & \text{otherwise} \end{cases}$$

With $\mathbf{G}$ being the set of representations conforming to the FOAF profile (cf. Listing 4.1).

Then the request $q \in \mathcal{Q}$ is formed with the pair consisting of the identifier of *YoucTagh* $i_{yt}$ and the constraint set $\mathbf{C}$. Formally:

$$q = \langle i_{yt}, \mathbf{C} \rangle$$

Here also, we can use the CN measure $m_{\text{pmax}}$ to compute the final acceptability values.

After that, we can define the server data $s_{\text{yt}}$ as follows:

$$s_{\text{yt}} = \langle \mathbf{D}_{\text{yt}}, m_{\text{pmax}}, \mathbf{C}_s \rangle$$

With $\mathbf{D}_{\text{yt}}$ a set containing the three RDF representations of *YoucTagh*, and $\mathbf{C}_s$ the server's preferences defined as follows:

1. `<me/yt-foaf.ttl>`: 0.7
2. `<me/yt-foaf-org.ttl>`: 0.9
3. `<me/yt-schema.ttl>`: 0.8

Finally, we define the web server $w_{\text{yt}}$, which maps $i_{yt}$ to the server data $s_{\text{yt}}$. Formally:

$$w_{\text{yt}} : i_{yt} \mapsto s_{\text{yt}}$$

The content negotiation process as defined in Definition 4.9 would then select the FOAF RDF representation.

## 4.3 Discussions on CON-NEG Applicability

In this section we present two different applications of CON-NEG to demonstrate its practical applicability. The first application focused on the Apache server [191] discussed in Section 4.3.1. We show how our model captures the essential concepts of the content negotiation process that allows a successful negotiation with this server, and how the different concepts are implemented. The second application is our implementation, where we used the OOP paradigm and transformed the different concepts into classes and interfaces (Section 4.3.2). We simulate the use case presented in Section 4.1.

### 4.3.1 CON-NEG Applied to HTTP and Apache Server

To run the content negotiation classic dimensions scenario in an Apache server, we use: a *type-map file*, some *HTTP headers* and a *negotiation algorithm* [191].

The type-map file, contains what we consider to be representation descriptions of a resource. An example of such a file is given in Listing 4.2.

Listing 4.2: The type-map file used to serve the resource *Abies numidica*

```
1  URI: abies_numidica
2
3  Content-type: text/html; qs=1
4  URI: abies_numidica.html
5
6  Content-type: application/pdf; qs=0.8
7  URI: abies_numidica.pdf
8
9  Content-type: text/plain; qs=0.6
10 Content-language: fr
11 URI: abies_numidica.fr.txt
12
13 Content-type: text/plain; qs=0.7
14 Content-language: en
15 URI: abies_numidica.en.txt
16
17 Content-type: application/rdf+xml; qs=0.75
18 URI: abies_numidica.rdf
19
20 Content-type: text/turtle; qs=0.85
21 URI: abies_numidica.ttl
```

The first line defines the *resource* of interest. Note that all IRIs are relative to the type-map file, we assume that the file is located at the IRI <species/abies_

numidica.var>

Then we describe all the available *representations*. At the same time, we define the server's constraints using the quality-of-source parameter `qs`.

Similarly, the HTTP `accept-*` headers are used to express and convey the client's constraints. The following headers define the preferences of the various requests expressed in the scenario presented in Section 4.1.1:

1. `accept: application/rdf+xml`
2. `accept: text/turtle, application/n-triples, application/rdf+xml, application/trig, application/n-quads, application/ld+json`
3. `accept: image/*, text/plain; q=0.8`
4. `accept-language: fr`
5. `accept-language: ar, en; q=0.8`
6. `accept: text/*; q=0.2, text/turtle`
7. `accept: text/*, text/turtle; q=0.2`

Note that media types, language tags and wildcard values are used to define a subset of representations and assign preferred acceptability values to them.

A HTTP `GET` request can then be formed by combining the *Abies numidica* IRI with the appropriate constraints.

The CN measure is defined by the negotiation algorithm "Multiply the quality factor from the Accept header with the quality-of-source factor for this variant's media type, and select the variants with the highest value".

The algorithm and the type-map file define the server data in our abstract model, which is considered in a negotiation process by the Apache web server.

### 4.3.2 CON-NEG Applied to OOP

In our implementation, we use the OOP paradigm. First, we model the various concepts as classes and interfaces. Figure 4.1 shows the class diagram of the general abstract model. The implementation is openly available on Github.[4]

The class `FirstScenario` simulates the scenario presented in Section 4.1.1. It consists of several parts, which we will introduce in the following paragraphs.

[4]CN Formalisation repository: https://github.com/youctagh/cn-formalisation

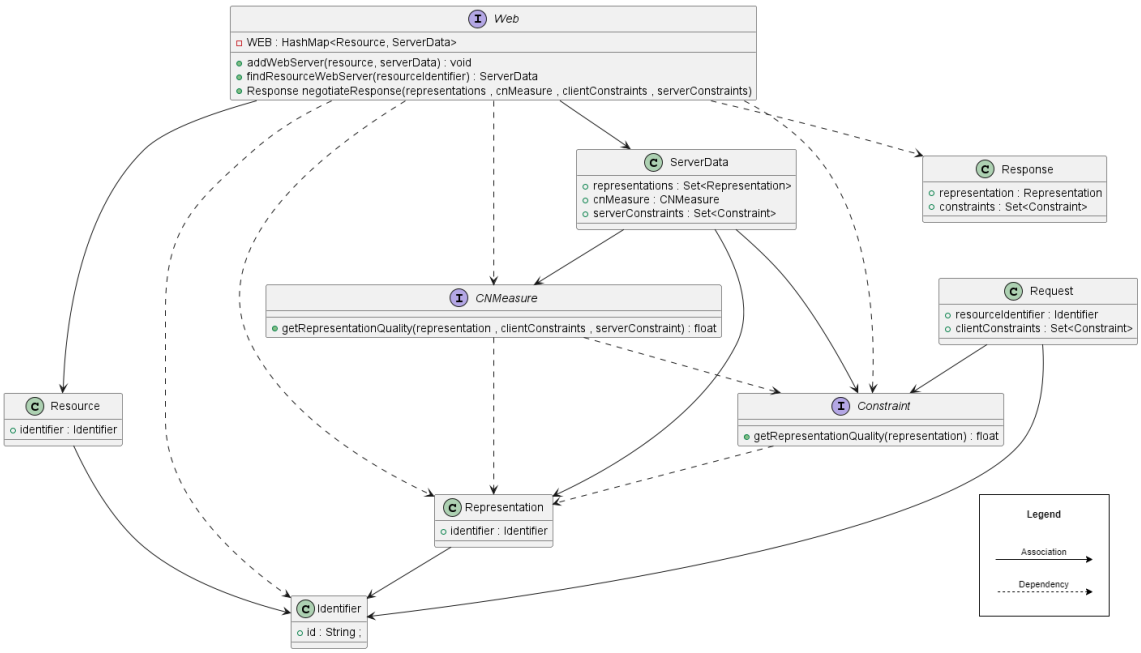



Figure 4.1: A class diagram of the general abstract model

First, we define the *resource* of interest and its various *representations*. To identify them, we use the function *id*, as shown in the snippet in Algorithm 1.

**Algorithm 1:** The creation of a resource and a representation

1 Identifier $i_{an}$ = <http://youctagh.com/species/abies_numidica>
2 Resource abiesNumidica = `id`($i_{an}$)
3 ...
4 Identifier $i_{d_1}$ = <http://youctagh.com/species/abies_numidica.html>
5 Representation $d_{an\_html}$ = `id`($i_{d_1}$)

Second, we define the server's constraints as shown in Algorithm 2. The constraint function takes a representation as input and assigns an acceptability value in the range $[0, 1]$. For instance, if the representation is $d_{an\_html}$, we assign the acceptability value 1.

**Algorithm 2:** The creation of a server's constraint $c_s$

**input :** A representation $d \in \mathcal{D}$
**output:** An acceptability value in the range $[0, 1]$

```
1  if d = d_an_html then
2      return 1
3  else if d = d_an_pdf then
4      return 0.8
5  else if d = d_an_txt_fr then
6      return 0.6
7  else if d = d_an_txt_en then
8      return 0.7
9  else if d = d_an_rdf_xml then
10     return 0.75
11 else if d = d_an_rdf_turtle then
12     return 0.85
13 else
14     return 0
```

Similarly, we can define the client's constraints. A snippet of the constraint function is shown in Algorithm 3 to get the constraint for the request *(5) I prefer an Arabic representation. If none is available, give me an English one.* Note that in this algorithm we rely on URI prefixes to define a subset of representations and assign preferred acceptability values to them. The function endsWith checks whether a URI ends with a given string.

**Algorithm 3:** The creation of the client's constraint $c_5$

**input :** A representation $d \in \mathcal{D}$
**output:** An acceptability value in the range $[0, 1]$

```
1  Identifier i such that id(i) = d
2  if endsWith(i,".ar") then
3      return 1
4  else if endsWith(i,".en") then
5      return 0.8
6  else
7      return 0
```

A request can then be formed by combining the *Abies numidica* URI with the

appropriate constraints, as shown in Algorithm 4.

**Algorithm 4:** The creation of a request

```
1 C_c = {c_5}
2 Request q = ⟨i_an, C_c⟩
```

In our implementation, the CN measure $m_{\text{pmax}}$ calculates the final acceptability value by multiplying the maximum value of the client's constraints by the maximum value of the server's constraints, as shown in Algorithm 5.

**Algorithm 5:** The creation of a CN measure

**input :** A representation $d \in \mathcal{D}$, a set of client constraints $\mathbf{C}_c$ and a set of server constraints $\mathbf{C}_s$.
**output:** An acceptability value in the range $[0, 1]$

```
1  float maxClientQuality = 0, clientQuality = 0
2  foreach c ∈ C_c do
3      clientQuality = c(d)
4      if maxClientQuality < clientQuality then
5          maxClientQuality = clientQuality
6  float maxServerQuality = 0, serverQuality = 0
7  foreach c ∈ C_s do
8      serverQuality = c(d)
9      if maxServerQuality < serverQuality then
10         maxServerQuality = serverQuality
11 return maxClientQuality × maxServerQuality
```

The `Web` interface simulates the Web and provides several utility functions:

1. `addWebServer`, which adds a new web server to the Web. In practice, this can be seen as registering a new domain.

2. `findWebServer` which, given a resource identifier, redirects to the web server that manages that identifier.

3. `negotiate` which, given a request for a resource, negotiates an acceptable representation.

**Algorithm 6:** The creation of the server data $s_{\text{an}}$

$\mathbf{D}_{\text{an}} = \{d_{an\_html}, d_{an\_pdf}, d_{an\_txt\_fr}, d_{an\_txt\_en}, d_{an\_rdf\_xml}, d_{an\_rdf\_turtle}\}$
$\mathbf{C}_s = \{c_s\}$
$s_{\text{an}} = \langle \mathbf{D}_{\text{an}}, m_{\text{pmax}}, \mathbf{C}_s \rangle$

Algorithm 6 shows a snippet of the server data $s_{\text{an}}$, and Algorithm 7, shows a snippet of the negotiation process of the *Abies numidica* resource.

**Algorithm 7:** The negotiation of the *Abies numidica* resource

**input :** A request $q_{\text{an}} = \langle i_{an}, \mathbf{C}_c \rangle$
**output:** A representation that maximises the server's CN measure

$w_{\text{an}}$ = `findWebServer(`$i_{an}$`)`
$\langle \mathbf{D}_{\text{an}}, m_{\text{pmax}}, \mathbf{C}_s \rangle = w(i_{an})$
Representation bestRepresentation = $d_{\text{no\_rep}}$
float bestQuality = 0, representationQuality = 0
**foreach** $d \in \boldsymbol{D}_{an}$ **do**
    $representationQuality = m_{\text{pmax}}(d, \mathbf{C}_c, \mathbf{C}_s)$
    **if** $representationQuality > bestQuality$ **then**
        bestQuality = representationQuality
        bestRepresentation = representation
**return** bestRepresentation

The class `SecondScenario` simulates the scenario presented in Section 4.1.2. First, we add some objects to our implementation as shown in Figure 4.2.

Then we define the different concepts as in the first scenario. However, here we use the `RDFRepresentation` and `Profile` as shown in the snippet in Algorithm 8. At this stage we use the function `conformsTo` to assert that a RDF representation conforms to a profile.

**Algorithm 8:** The creation of a resource, an RDF representation and a profile

Identifier $i_{yt}$ = `<http://youctagh.com/me/>`
Resource youctagh = `id(`$i_{yt}$`)`
Identifier $i_{\text{p\_foaf}}$ = `<http://youctagh.com/me/profiles/prof-foaf.ttl>`
Profile ProfFoaf = `id(`$i_{p\_foaf}$`)`
Identifier $i_{\text{yt\_foaf}}$ = `<http://youctagh.com/me/yt-foaf.ttl>`
RDFRepresentation $g_{\text{yt\_foaf}}$ = `id(`$i_{yt\_foaf}$`)`
`conformsTo(`$g_{yt\_foaf}$, ProfFoaf`)`
...

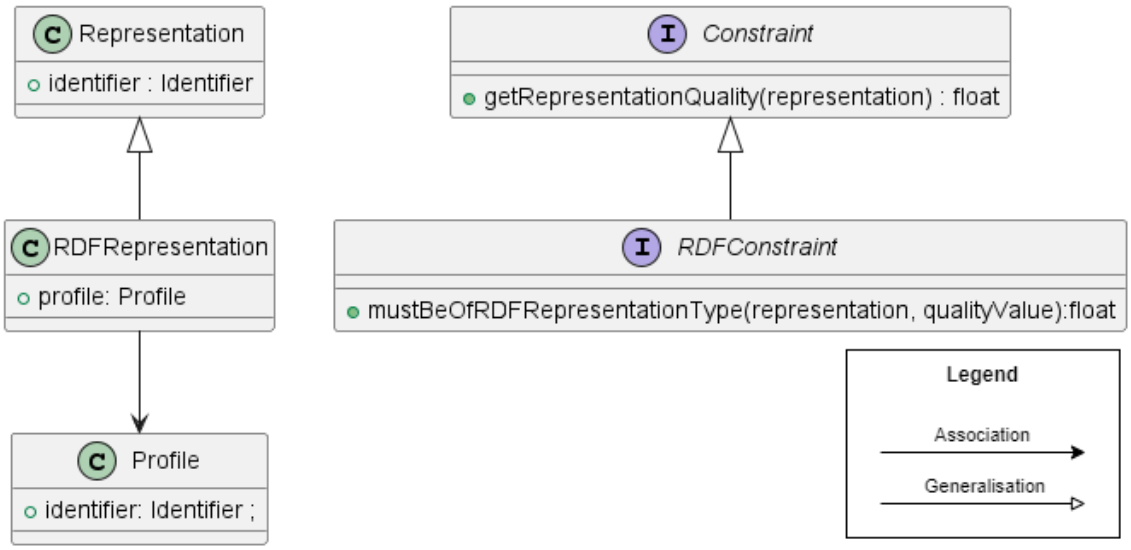



Figure 4.2: A class diagram of the formalisation extension to semantic context

After that we can create the server constraint (see Algorithm 9). Note that here we have to check that the representation is an RDF representation, otherwise its acceptability value is reduced to 0 (see lines 1-2). Furthermore, our implementation uses a *simple string comparison* [20, Section 6.2.1]. This means that to assign an acceptability value we compare two given profile URIs character by character. However, this style of constraint validation has its shortcomings, which we address in Chapter 5.

**Algorithm 9:** The creation of a server's constraint $c_s$ while enforcing the RDF representation type

**input :** A representation $d \in \mathcal{D}$
**output:** An acceptability value in the range $[0, 1]$

```
if d ∉ D_g then
    return 0
else if i_p_foaf = i_p such that id(i_p) = p ∧ conformsTo(d, p) then
    return 0.7
else if i_p_foaf_org = i_p such that id(i_p) = p ∧ conformsTo(d, p) then
    return 0.9
else if i_p_schema = i_p such that id(i_p) = p ∧ conformsTo(d, p) then
    return 0.8
else
    return 0
```

## 4.4 Summary

Despite the fact that content negotiation is of great utility, no formalisation has been done to specify its behaviour. In this chapter, we presented the CON-NEG model for content negotiation. We started by describing a general case that reflects current practice and showed how it fits into the HTTP protocol's way of specifying content negotiation in media type and language dimensions (Section 4.2.1). We then showed how we can extend the general model in a semantic context to allow the negotiation of RDF graphs conforming to a profile (Section 4.2.2). In Section 4.3 we discussed the applicability of our model in two contexts. The first one using the Apache server (cf. Section 4.3.1). The second using our own implementation that uses the OOP paradigm (cf. section 4.3.2). CON-NEG is our contribution to address Research Question 1: *How can we formalise the problem of content negotiation in a general and extensible way?*.

Chapter 5

# Towards a Flexible and Relaxed Semantic Content Negotiation

## Contents



In the previous chapter we introduced CON-NEG, our model for content negotiation. We saw how it fits with the current state of achieving content negotiation in the classical dimensions and how it could be extended to the profile one. In this chapter we continue our modelling towards capturing flexible and relaxed semantic content negotiation in order to address Research Question 2.

In Section 5.1, we start by extending the use case presented in Section 4.1 to illustrate the need for both flexible and relaxed negotiation. Then, in Section 5.2 we describe how we can use semantic validation to enhance the content negotiation process while being guided by the content negotiation stages (cf. Section 3.1): first, by indicating hard and soft constraints; then, by relaxing the validation and reporting finer validation results; and finally, by interpreting the validation report. In Section 5.3 we present S-CON-NEG (for Semantic CON-NEG), our model to formalise

such flexible and relaxed content negotiation. In Section 5.4 we discuss the applicability of our proposal by extending the implementation presented in Section 4.3.

## 5.1 Incremental Use Case - Part II

In Section 4.1.2 we considered a general case of semantic content negotiation: the client requests a resource representation that must conform to a profile from a set of profiles already known by the server (the profile specifies some semantic and/or structural constraints). However, one of the research objectives is to have a negotiation that has the additional properties of being flexible and relaxed. Moreover, it should be guided by the content negotiation stages (cf. Section 3.1).

To better frame the considered scenarios, we build on the incremental use case presented in Section 4.1 and consider two scenarios. In Section 5.1.1 we consider the case of negotiation with on-the-fly validation. In Section 5.1.2 we cover the case of finer and optional constraints with relaxed validation.

### 5.1.1 CN in a Semantic Context with On-The-Fly Validation

*YoucTagh* informed his friend *Lyncis* about the representations he had created to introduce himself as the creator of the resources served by his server. *Lyncis* liked the idea and did the same on her server. She created two representations to introduce herself: the first uses only `Schema.org`, exactly the same profile as *YoucTagh*'s; the second uses `FOAF` in addition to `Dublin Core` instead of `Organisation`.

The RDF representations and their corresponding profiles are as follows:

- The `FOAF + Dublin Core` representation present at: `http://lyncis.com/me/lyncis-foaf-dc.ttl` conforms to the profile `http://lyncis.com/profiles/prof-foaf-dc.ttl`
- The `Schema.org` representation at: `http://lyncis.com/me/lyncis-schema.ttl` conforms to the profile `http://lyncis.com/profiles/prof-schema.ttl`

If *Lyncis* negotiates with her server for a representation matching the profile identified by `http://lyncis.com/profiles/prof-schema.ttl`, she will be able to get the correct representation (i.e. the one identified by `http://lyncis.com/me/lyncis-schema.ttl`). However, using the same IRI to negotiate with *YoucTagh*'s server will not yield any representations, even though the representation `http://youctagh.com/me/yt-schema.ttl` exists. Similarly, requesting a representation that conforms to the profile `http://lyncis.com/profiles/prof-foaf-dc.ttl` will

not return a representation. However, looking at the representation at http://youctagh.com/me/yt-foaf-org.ttl (cf. Listing 5.1), it conforms to the requested profile shown in Listing 5.2. This profile states that:

1. at least one instance of the class `foaf:Person` must be present in the graph (lines 1-6);
2. all instances of type `foaf:Person` must have at least at least one path `dct:creator` and one path `dct:title` (lines 7-16).

Listing 5.1: A snippet of *YoucTagh*'s `FOAF + Org` representation

```
1  yt:youctagh a foaf:Person;
2          foaf:name "Youc Tagh"@en;
3          dct:title "Worker";
4          dct:creator  <http://youctagh.com/me>;
5          org:memberOf yt:company .
6
7  yt:company a org:Organisation;
8          skos:prefLabel "YoucTagh's Company".
```

Listing 5.2: A snippet of the `FOAF + Dublin Core` profile expressed with SHACL

```
1  ytshape:FoafPersonMinCount a sh:NodeShape;
2   sh:targetNode foaf:Person;            # At least
3   sh:property [                         # one instance
4    sh:path [sh:inversePath rdf:type ;]; # of foaf:Person must
5    sh:minCount 1;                       # exist in the graph
6   ].
7  ytshape:PersonShape  a sh:NodeShape;
8   sh:targetClass foaf:Person; # all instances of type foaf:Person
9   sh:property [
10   sh:path dct:creator;       # must have at least
11   sh:minCount 1;             # one path dct:creator
12  ];
13  sh:property [               # and
14   sh:path dct:title;         # must have at least
15   sh:minCount 1;             # one path dct:title
16  ].
```

### 5.1.2 CN in a Semantic Context with Finer Constraints

Now *YoucTagh* knows that he should add the functionality to consider previously unknown profiles and perform on-the-fly validation to select the appropriate representation to deliver. However, he does not want to have only an *all-or-none* way

```
yt:youctagh a foaf:Person;
 foaf:familyName "Tagh"@en;
 foaf:givenName "Youc"@en;
 foaf:knows ly:lyncis.

ly:lyncis a foaf:Person;
 foaf:familyName "Lyncis"@en;
 foaf:givenName "Nina"@en.
```

(a) Representation 1 "$R_1$"

```
yt:youctagh a foaf:Person;
 foaf:name "YoucTagh"@en;
 foaf:mbox <mailto:youc@yt.com>;
 foaf:knows ly:lyncis.

ly:lyncis a foaf:Person;
 foaf:name "Nina Lyncis"@en.
```

(b) Representation 2 "$R_2$"

```
yt:youctagh a foaf:Person;
 foaf:familyName "Tagh"@en;
 foaf:givenName "Youc"@en;
 foaf:mbox <mailto:youc@yt.com>;
 foaf:knows ly:lyncis.

ly:lyncis a foaf:Person;
 foaf:familyName "Lyncis"@en;
 foaf:givenName "Nina"@en;
 foaf:mbox <mailto:nina@ly.com>.
```

(c) Representation 3 "$R_3$"

```
yt:youctagh a foaf:Person;
 foaf:familyName "Tagh"@en;
 foaf:givenName "Youc"@en;
 foaf:mbox <mailto:youc@yt.com>;
 foaf:knows ly:lyncis.

ly:lyncis a foaf:Person;
 foaf:familyName "Lyncis"@en;
 foaf:givenName "Nina"@en.
```

(d) Representation 4 "$R_4$"

Multiple Listing 5.1: Four snippets of four representations of the resource *YoucTagh*

of validating profiles. He wants to be able to handle finer optional preferences with the following semantics: all representations that validate the required constraints are acceptable. However, the acceptable representation that validates the most optional constraints is preferred.

For example, suppose we have the following profile: the representation (1) must have at least one `foaf:Person` instance; (2) all instances of `foaf:Person` must contain one `foaf:familyName` and one `foaf:givenName` predicate. *Preferably*, each `foaf:Person` would have a `foaf:mbox` predicate. With the four representations shown in Multiple Listing 5.1.[1] The expected results are:

- $R_1$, $R_3$ and $R_4$ are all acceptable representations. However, $R_1$ is the least acceptable representation, since it validates only the mandatory preferences;

[1]We assume the representations are identified by URI `http://youctagh.com/me/yt-representation-{x}.ttl` with `x` being 1, 2, 3 or 4.

- $R_4$ is in second place because only the person `yt:youctagh` has a `foaf:mbox` predicate.
- $R_3$ is preferred, since in addition to fulfilling all the mandatory preferences, all `foaf:Person` have a `foaf:mbox`;
- $R_2$ is not valid because the person `yt:youctagh` has neither `foaf:givenName` nor `foaf:familyName` predicates;

## 5.2 Content Negotiation Process Enhancement

To achieve the above two scenarios, we need to enhance the content negotiation process by providing a way to request *finer* constraints, *relax* their validation and be able to report the result for interpretation. In this pursuit we have been guided by the different stages of content negotiation. This allows us to preserve the *orthogonality* principle and to improve each stage separately. Since we are focusing on semantic validation in this chapter, we will not discuss the first stage (i.e. discovery), which is discussed in details in Chapter 7. Each of the other stages will have its own dedicated discussion to explore how semantic validation would improve the content negotiation process:

- In Section 5.2.1, we show how a client can specify finer constraints to be considered by the server. Thus we enhance the *request formulation* stage.
- In Section 5.2.2, we show how a server can relax the validation and selection of the representation given the finer preferences. Thus we enhance the *selection/adaptation* stage.
- In Section 5.2.3, we show how a server can report the relaxed validation. Thus we enhance the *response indication* stage.
- In Section 5.2.4, we show how a client should react to a response from a server that has gone through this kind of enhanced content negotiation. Thus we enhance the *response interpretation* stage.

In what follows, we use SHACL as a showcase technology. The same ideas can be transferred and adapted to other semantic validation languages such as ShEx. We selected SHACL because of its wide acceptance as a W3C recommendation and because the SHACL vocabulary defines the `sh:severity` property, which is instrumental for our purposes.

### 5.2.1 Indicating Hard and Soft Constraints

Chapter 4 focuses on enabling clients to request a representation that conforms to a profile. But the set of acceptable profiles was finite and already known to the server. Nevertheless, using semantic validation languages (e.g. SHACL), clients can define specific profiles to be validated on-the-fly by a validation engine. On-the-fly validation is useful for dealing with previously unknown profiles, thus giving the client the ability to request custom profiles.

To convey the preferred profile, the same technique of pointing to it with a URI can be used, except that in this context it must be dereferenceable (e.g. to a SHACL shape graph). Other alternatives are to send the profile content itself in the request (either *plain* or encoded in *base64* for example). This approach is beneficial for reducing network traffic, as servers do not have to make additional requests to get the requested profiles. However, if the profile is very large, such a technique inherits the size drawback discussed in Section 3.3.3. These approaches are used when formulating a request for some existing third-party SHACL validators, for example: AzureWebSite,[2] Corese,[3] Interoperability Test Bed,[4] RDF Plaground,[5] RDF Shape,[6] and Sparna.[7]

Using semantic validation languages, we are able to define finer constraints. However, constraint precedence (i.e. priority and ordering) is still applied at the profile level. For example, the quality values used to convey the preferred SHACL profiles assign a relative weight to the whole profile, despite the fact that a SHACL shape graph can contain multiple shapes, and that some are mandatory and others are not. This aspect was demonstrated by the scenario in Section 5.1.2. This means that not all constraints have to be defined as hard constraints.

In SHACL, the property `sh:severity` can be used to satisfy the two requirements: ordering of finer constraints and definition of preferred optional ones. All shapes with severity `sh:Warning` or `sh:Info` are considered to define soft constraints, while those with `sh:Violation` or not defining any are considered to define hard constraints. At the same time, in the context of content negotiation, hard constraints can be used as a filtering feature (i.e. is a representation acceptable or not), while soft constraints can be used for ordering (i.e. the more a representation violates soft constraints, the lower it is ranked, and shapes with the warning severities are more

[2] https://shacl.azurewebsites.net/validate
[3] https://corese.inria.fr/service/shape
[4] https://www.itb.ec.europa.eu/shacl/any/upload
[5] http://rdfplayground.dcc.uchile.cl/api/shape/shacl_isvalid
[6] https://api.rdfshape.weso.es/api/schema/validate
[7] https://shacl-play.sparna.fr/play/validate

important than the info ones).

Note that such a property for expressing severities is not defined in ShEx. However, it could be added as a shape annotation [137, Section 5.9] and then used in the validation result [67, Section 7.7].

Listing 5.3 shows an example of a SHACL profile that expresses the requirements of the scenario in Section 5.1.2 through the use of severities.

Listing 5.3: A snippet of a SHACL profile defining hard and soft constraints

```
ytshape:FoafPersonMinCount a sh:NodeShape;
    sh:targetNode foaf:Person;
    sh:property [
        sh:path [sh:inversePath rdf:type ;];    sh:minCount 1;
        sh:severity sh:Violation ;
    ].
ytshape:PersonNameShape  a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:familyName ;  sh:minCount 1 ;
        sh:severity sh:Violation ;
    ];
    sh:property [
        sh:path foaf:givenName ;  sh:minCount 1 ;
        sh:severity sh:Violation ;
    ].

ytshape:PersonMailShape  a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:mbox ;  sh:minCount 1 ;
        sh:severity sh:Warning ;
    ];
```

To achieve a finer ordering of SHACL shapes, a custom weight predicate can be used at the shape level, e.g. `shy:weight`. Note that this predicate is not part of the official standard. Listing 5.4 gives an example of the use of such a property. Two shapes, one with a weight of 1 and the other with a weight of 2, mean that the latter is twice as important as the former.

Listing 5.4: A snippet of a SHACL profile defining shape ordering through weights

```
ex:mailShape  a sh:NodeShape ;
    sh:targetClass foaf:Person ;
    sh:property [
        sh:path foaf:mbox ;  sh:minCount 1 ;
        sh:severity sh:Warning ;
```

```
6           shy:weight 2 ;
7       ];
8
9   ex:shaMailShape  a sh:NodeShape ;
10      sh:targetClass foaf:Person ;
11      sh:property [
12          sh:path foaf:mbox_sha1sum ;  sh:minCount 1 ;
13          sh:severity sh:Warning ;
14          shy:weight 1 ;
15      ];
```

A server that is able to handle both flexible (hard and soft constraints) and strict (hard constraints only) content negotiation can provide the ability to select a conformance checking mode via an additional preference. For example, one can define the `accept-conformance` header with the values `strict` or `flexible`.

Taken together, the above techniques would allow a client to indicate acceptance of representations that are partially compliant with a profile, and consequently engage in relaxed content negotiation.

### 5.2.2 Relaxing Validation

Once the client has expressed and communicated its preferences for flexible and relaxed content negotiation, the server should be adapted to consider these types of preferences.

First, to enable on-the-fly validation, the server should not rely on simple string comparison of URIs (i.e. comparing two URIs character by character). However, the server must perform a conformance check and decide according to the obtained validation results. In the case of SHACL, if the boolean value of the `sh:conforms` property in the validation report is `true`, this means that the RDF representation conforms to the profile. To relax the process, the server would also accept RDF representations that violate soft constraints. In SHACL, RDF representations with a validation report containing only violations of shapes with severities `Info` or `Warning` are considered acceptable.

In relaxed content negotiation, it is expected that more representations will validate the requested profile. The server can respond with a `300 Multiple Choices` status code, so that a client can engage in reactive content negotiation. However, in the case of a proactive style, where the server needs to select the most appropriate representation, a scoring function can be used to rank the acceptable alternatives. A first example of such a scoring function is to penalise the representations that violate soft constraints. An example of such a function is provided in Listing 13.

A second scoring function that can be used to select among several acceptable RDF representations is knowledge density, i.e. selecting the representation with the highest number of triples. However, this assumes that more triples imply more knowledge for the client, which is not always true.

So far we have assumed that the server has an acceptable representation. However, sometimes none of the available representations are valid. In this case, a response with the `406 Not Acceptable` status code can be used. A second option is to send a predefined default representation in a response with `200 OK` status code. A third option is to send a representation that violates the minimum number of constraints (both hard and soft). A scoring function that can be used is the ratio of valid constraints to tested constraints: If $v_c$ is the number of valid constraints and $n_c$ is the number of tested constraints, we select the representation that has the highest `ratio` score calculated with the formula:

$$\texttt{ratio} = \frac{v_c}{max(1, n_c)}$$

Note that $max(1, n_c)$ is used to avoid dividing by 0, the case of empty graphs for example.

### 5.2.3 Reporting Finer Validation

At the *response indication* stage, the server needs to indicate the response to the client. A `Link` header can be used in conjunction with the `profile` relation type to indicate the profiles to which the returned or available representations conform, as seen in Section 3.4.2. However, when the negotiation is finer and more relaxed, we need to refine the response to convey the result of the validation. There are two approaches to achieve this, each of which will have an impact on the next stage of interpreting the response, as we discuss in Section 5.2.4.

In the first approach, the server can create a new profile based on the requested one, but discard any constraints that the representation does not satisfy. Then, the same approach using the `link` header and the `profile` relation type can be applied.

In the second approach, the server sends back an unmodified profile using the `Link` header and `profile` relation type, in addition to the resulting report of the validation of that profile against the selected representation. The provided report is then used by the client to gain insights and understanding of the returned response.

When using SHACL, a standard validation report as defined in the W3C recommendation can be used [97, Section 3.6]. If necessary, this report can be extended to provide additional details. For example, the *DASH Data Shapes Vocabulary* [95,

Section 5] extends the `sh:AbstractResult` to represent different types of validation results. Similarly, a recent work on a framework includes and exploits probabilistic information in SHACL validation reports by enriching the result with additional metrics [57].

Likewise, we identified four options for transmitting validation reports:

First, by extending the variant description [84, Section 5] by using the `report` *extension-attribute* with the validation report URI as value, for example:

```
Alternates: {"rep.1" 0.7 {type text/turtle}
             {report <http://www.uri-to-report.org>}}
```

Second, by using the `Link` header with a new relation type. We propose the relation `validation-report`, for example:

```
Link:  <http://www.uri-to-report.org>; rel="validation-report";
type="text/turtle"
```

Third, by including only the report in the body of the response. A special media type can be used, for example `application/problem+json` as proposed in the problem details approach [130], for example:

Listing 5.5: A snippet of a validation report using the problem details approach

```
{
 "type": "http://example.com/validation-report",
 "title": "No representation available that conforms to the
    requested profile.",
 "profile": "http://uri-to-requested-profile.com/"
 "errors": [{
    "detail": "details from the validation engine",
    "pointer": "http://uri-to-some-invalid-shape.com/"
  }]
}
```

Fourth, by sending both the representation and its validation report in the body of the response. A special multipart media type can be used, we propose the media type `multipart/validation-report`, for example:

Listing 5.6: A snippet of a response including both the representation and the validation report

```
...
Content-Type: multipart/validation-report; boundary=SEPARATION
```

```
--SEPARATION
Content-Type: text/turtle
... The representation ...
--SEPARATION
Content-Type: application/problem+json
... The validation report ...
--SEPARATION--
```

### 5.2.4 Interpreting Validation Report

In the final content negotiation stage i.e. *response interpretation*, the client interprets the response received and reacts accordingly. The HTTP method, the response status code, and the headers, all contribute to the semantics of the response, as explained in Section 2.3.2. However, in order to achieve flexible and relaxed content negotiation, we have to adapt the response as shown in Section 5.2.3. Below we describe how a client should behave in different situations.

If the response contains only a `Link` header containing one web link with the `profile` relation type, the client can understand that the provided representation conforms to the profile target of that web link. If it is the same identifier, the client knows that all constraints are satisfied. In the opposite case, this means that the provided representation does not satisfy some or all of the preferred constraints.

If the response contains a web link with the `profile` relation type identifying the same requested profile, the client needs to check whether a validation report has been transmitted. This can be done in four ways that we describe in the following paragraphs. Note that the first two provide only a URI to the report, which means that an additional dereferencing request is required which is not the case in the last ones:

- An `Alternates` header [84, Section 8.3] containing the variants descriptions using the `report` *extension-attribute* with the report URI as value.
- A web link with the relation type `validation-report` pointing to the report URI.
- `406 Not Acceptable` status code and a `Content-type` header with the value of a *problem details* media type, e.g. `application/problem+json`. and a `Location` header pointing to the representation that generated the report in the body, so that the client can dereference it if necessary.

- `200 OK` status code and a `Content-type` header with the value of the media type `multipart/validation-report` and a `boundary` parameter. The first occurrence of the boundary indicates the start of the representation. The second occurrence delimits the end of the representation and the start of the validation report. The third and final occurrence indicates the end of the report.

Table 5.1 provides a summary of the content negotiation enhancements proposed in this section.

| CN Stage | Enhancement | Enhancement Overview |
|---|---|---|
| Formulation of the request | Indicating hard and soft constraints | • use `sh:severity` *Violation*, *Warning* and *Info* to indicate the constraint type • use `shy:weight` to indicate the soft constraint weight • use the `accept-conformance` with either the value *strict* or *flexible* to indicate the conformance checking mode. |
| Selection and/or adaptation | Relaxing the validation | • Accept the representations that violate soft constraints • Use scoring functions to rank representations according to their validation reports using: (1) constraint weights; (2) knowledge density; (3) the ratio of valid constraints to tested constraints. |
| Indication of the response | Reporting finer validation | • use a `Link` header and a `profile` relation type to a profile that the provided representation conforms to • use a `Link` header and a `profile` relation type to a profile that the provided representation does not conform to, plus a validation report conveyed in: (1) a `report` *extension-attribute* in the `Alternates` header; (2) a web link with the `validation-report` relation type; (3) the response body that conforms to the *problem details* syntax; (4) a `multipart/validation-report` response where the first part is the representation and the second is the validation report. |
| Interpretation of the response | Interpreting validation reports | React according to the previous stage, if a validation report is not sent, this means that the representation conforms to the profile, otherwise detect which of the 4 options is used to send the report and interpret it. If multiple reports are sent, the server expects the client to make the selection. |

Table 5.1: A summary of the content negotiation enhancements proposed in Section 5.2

## 5.3 S-CON-NEG – A Model for Flexible and Relaxed CN

In this section we formalise the flexible and relaxed content negotiation. To do this, we reuse the structures defined in Chapter 4 and redefine them in this context. As an illustration, we take the scenario described in Section 5.1.2. Again, we structure our discussion in a bottom-up manner, introducing new formal definitions as needed.

In this scenario, the *resource* of interest is *YoucTagh*, which is *identified* by the IRI $i_{yt}$.

Here we have four different *RDF representations*, also identified by IRIs.

To accommodate on-the-fly validation and not just rely on simple string comparison of URIs, we further define the profile concept. But first we need to define two concepts. The first concept is *constraint type*, and the second is *shape*:

**Definition 5.1: Constraint Type**

We define the set of *constraint types* $\mathcal{T}$. A constraint type $t \in \mathcal{T}$ can take one of two values: $\mathsf{hard}$ to indicate that the constraint is a hard constraint, and $\mathsf{soft}$ to indicate that it is a soft one, formally:

$$\mathcal{T} = \{\mathsf{hard}, \mathsf{soft}\}$$

**Definition 5.2: Shape**

We define the set of all *shapes* $\mathcal{SH}$. A shape $sh \in \mathcal{SH}$ is a pair formed by a constraint $c \in \mathcal{C}$ and a constraint type $t \in \mathcal{T}$, formally:

$$\mathcal{SH} = \mathcal{C} \times \mathcal{T}$$

Now we can define the profile concept:

**Definition 5.3: Profile**

The set of all profiles is $\mathcal{P}$. A profile $p \in \mathcal{P}$ is a set of shapes, formally:

$$\mathcal{P} = 2^{\mathcal{SH}}$$

In our scenario, the client requests the profile $p_c \in \mathcal{P}$ with three shapes $sh_1, sh_2, sh_3 \in \mathcal{SH}$:

$$p_c = \{sh_1, sh_2, sh_3\}$$

The shapes are defined as follows:

(1) $sh_1$ is the pair $\langle c_1, \mathsf{hard} \rangle$, where $c_1 \in \mathcal{C}$ assigns the value 1 to any RDF representation $g \in \mathcal{D}_g$ if it has at least one `foaf:Person` instance, and 0 otherwise.

(2) $sh_2$ is the pair $\langle c_2, \mathsf{hard} \rangle$, where $c_2 \in \mathcal{C}$ assigns the value 1 to any RDF representation $g \in \mathcal{D}_g$ if all of its `foaf:Person` instances have a `foaf:familyName` and a `foaf:givenName` predicate, and 0 otherwise:

(3) $sh_3$ is the pair $\langle c_3, \mathsf{soft} \rangle$, where $c_3 \in \mathcal{C}$ assigns the value 1 to any RDF representation $g \in \mathcal{D}_g$ if all of its `foaf:Person` instances have a `foaf:mbox` predicate, and decrease this value for each violation (a concrete example is to decrease the value by 0.1 with a maximum of 10 violations).

When a graph is validated against a shape using a validation engine, the output is a validation result.

**Definition 5.4: Validation Result**

We define the set of all *validation results* $\mathcal{VR}$. A validation result $vr \in \mathcal{VR}$ is a triple of a result weight between $[0, 1]$; a constraint type $t \in \mathcal{T}$; and a string message from the set of strings $\mathsf{String}$, formally:

$$\mathcal{VR} = [0, 1] \times \mathcal{T} \times \mathsf{String}$$

When a graph is validated against a shape, we say that we are performing a conformance check.

**Definition 5.5: Conformance Check**

We define the set of all *conformance checks* $\mathcal{CC}$. A conformance check $cc \in \mathcal{CC}$ is a function that assigns a validation result $vr \in \mathcal{VR}$ to a pair of an RDF representation $g \in \mathcal{D}_g$ and shape $sh \in \mathcal{SH}$, formally:

$$cc : \mathcal{D}_g \times \mathcal{SH} \rightarrow \mathcal{VR}$$

For example, the conformance checks of the RDF representations $R_1$, $R_2$, $R_3$ and $R_4$ (cf. Listing 5.1) with the shapes $sh_1$, $sh_2$ and $sh_3$ could yield the following validation results:

For the first RDF representation:

$vr_1 = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_2 = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_3 = \langle 0.8, \mathsf{soft}, \text{“Not valid. yt:youctagh and ly:lyncis are missing foaf:mbox”} \rangle$

For the second RDF representation:[8]

$vr_4 = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_5 = \langle 0, \mathsf{hard}, \text{“Not Valid. yt:youctagh is missing a foaf:familyName”} \rangle$
$vr_6 = \langle 0.9, \mathsf{soft}, \text{“Not valid. ly:lyncis is missing foaf:mbox”} \rangle$

For the third RDF representation:

$vr_7 = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_8 = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_9 = \langle 1, \mathsf{soft}, \text{“Valid”} \rangle$

For the fourth RDF representation:

$vr_{10} = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_{11} = \langle 1, \mathsf{hard}, \text{“Valid”} \rangle$
$vr_{12} = \langle 0.9, \mathsf{soft}, \text{“Not valid. ly:lyncis is missing foaf:mbox”} \rangle$

We call a client request for a profiled representation with relaxed constraints a *semantic request*.

**Definition 5.6: Semantic Request**

We define the set of all *semantic requests* $\mathcal{SR}$. A semantic request $sr \in \mathcal{SR}$ is a pair of an identifier with a set of profiles, formally:

$$\mathcal{SR} = \mathcal{I} \times 2^{\mathcal{P}}$$

In our running example, the client request $sr \in \mathcal{SR}$ is defined by the IRI $i_{yt}$ and the singleton $\{p_c\}$, formally:

$$sr = \langle i_{yt}, \{p_c\} \rangle$$

[8]Note that for $vr_5$, we are reporting only one violation. The message can also contain all validation results.

**Definition 5.7: Semantic CN Measure**

We define the set of all *Semantic CN measures* $\mathcal{SM}$. A semantic CN measure $sm \in \mathcal{SM}$ is a function that associates a pair consisting of the final acceptability value in the range $[0, 1]$, with the set of validation results $2^{\mathcal{VR}}$, to an RDF representation given two sets of profiles. The first set is given by the client in the request, and the second is the server's set of profiles, formally:

$$sm : \mathcal{D}_g \times 2^{\mathcal{P}} \times 2^{\mathcal{P}} \rightarrow [0, 1] \times 2^{\mathcal{VR}}$$

In our running example, we call $p_s$ the profile of the server to which all available RDF representations conform. $p_s$ indicates that there is at least one `foaf:Person` instance in the graph. This means that the obtained validation results for $R_1$, $R_2$, $R_3$, and $R_4$ are all: $\langle 1, \mathsf{hard}, \text{“Valid”} \rangle$.

We define the semantic CN measure $sm_{\text{pmax}}$. It calculates the final acceptability value by multiplying the two maximum weights of the validation results obtained when checking the conformance of the representation to the client's set of profiles and the server's set of profiles. The validation result set produced by $sm_{\text{pmax}}$ is the union of the two validation result sets with the maximum weights when performing the conformance check.

The $sm_{\text{pmax}}$ would produce the following results when considering the two profile sets $\{p_s\}$ and $\{p_c\}$

For the first RDF representation $R_1$:

$$sm_{\text{pmax}} : \langle R_1, \{p_s\}, \{p_c\} \rangle \mapsto \langle 0.8, \{vr_1, vr_2, vr_3, vr_{13}\} \rangle$$

For the second RDF representation $R_2$:

$$sm_{\text{pmax}} : \langle R_2, \{p_s\}, \{p_c\} \rangle \mapsto \langle 0, \{vr_4, vr_5, vr_6, vr_{14}\} \rangle$$

For the third RDF representation $R_3$:

$$sm_{\text{pmax}} : \langle R_3, \{p_s\}, \{p_c\} \rangle \mapsto \langle 1, \{vr_7, vr_8, vr_9, vr_{15}\} \rangle$$

For the fourth RDF representation $R_4$:

$$sm_{\text{pmax}} : \langle R_4, \{p_s\}, \{p_c\} \rangle \mapsto \langle 0.9, \{vr_{10}, vr_{11}, vr_{12}, vr_{16}\} \rangle$$

Note that the final acceptability values are consistent with what was expected in the Section 5.1.2.

**Definition 5.8: Semantic Server Data**

We define the set of all *semantic server data* $\mathcal{SS}$. A semantic server data $ss \in \mathcal{SS}$ is a triple consisting of a set of RDF representations, a semantic CN measure to compute the final acceptability values for those representations with the adequat report, and a set of profiles of that server. Formally:

$$\mathcal{SS} = 2^{\mathcal{D}_g} \times \mathcal{SM} \times 2^{\mathcal{P}}$$

For example, in our scenario, the server semantic data of the *YoucTagh* resource is the triple $ss_{\text{yt}}$ consisting of the set of four RDF representations $\mathbf{D}_{\text{yt}}$ and the semantic CN measure $sm_{\text{pmax}}$, and the singleton $\{p_s\}$. Formally:

$$\mathbf{D}_{\text{yt}} = \{R_1, R_2, R_3, R_4\}$$

$$ss_{\text{yt}} = \langle \mathbf{D}_{\text{yt}}, sm_{\text{pmax}}, \{p_s\} \rangle$$

**Definition 5.9: Semantic Web Server**

A *semantic web server* $sw$ is a partial function over the set of identifiers $\mathcal{I}$, which assigns to some identifiers the semantic server data of the resources identified by those identifiers, formally:

$$sw : \mathcal{I} \nrightarrow \mathcal{SS}$$

The set $\mathcal{SW}$ of all web servers corresponds to our model of the Web where all servers are semantically enabled. Note that only one semantic web server can serve any given identifier. Therefore the domains of the semantic web servers are pairwise disjoint: $\forall i \in \mathcal{I}, \forall sw_1, sw_2 \in \mathcal{SW}, i \in \mathcal{I}_{sw_1} \wedge i \in \mathcal{I}_{sw_2} \implies sw_1 = sw_2$

For example, in our scenario, *YoucTagh*'s semantic web server $sw_{\text{yt}}$ handles the IRI of the *YoucTagh* resource and maps $i_{yt}$ to the semantic server data $ss_{\text{yt}}$. Formally:

$$sw_{\text{yt}} : i_{yt} \mapsto ss_{\text{yt}}$$

**Definition 5.10: Semantic Content Negotiation Problem**

A *semantic content negotiation problem* can be defined as follows, given:

- A semantic web server $sw \in \mathcal{SW}$ on the Web serving resources identified

by some identifiers from the set $\mathbf{I}_s \subset \mathcal{I}$, such that for all $i \in \mathbf{I}_s$: $sw(i) \coloneqq \langle \mathbf{G}_i, sm_i, \mathbf{P}_{si} \rangle$

- A semantic request $sr \in \mathcal{SR}$ for a resource identified by an identifier $i_r \in \mathbf{I}_s$, with a set of profiles $\mathbf{P}_c \subseteq \mathcal{P}$. i.e. $sr \coloneqq \langle i_r, \mathbf{P}_c \rangle$

The semantic content negotiation problem aims to find an RDF representation with its validation results set such that:

$$\langle g_{\text{best}}, \mathbf{VR}_{\text{best}} \rangle =$$
$$\langle \operatorname*{argmax}_{g \in \mathbf{G}_i} (\{qv \in [0,1] \mid \exists \mathbf{VR} \subset \mathcal{VR}, sm_i(g, \mathbf{P}_c, \mathbf{P}_{si}) = (qv, \mathbf{VR})\}) \ ,$$
$$\operatorname*{argmax}_{\mathbf{VR} \subset \mathcal{VR}} (\{qv \in [0,1] \mid \exists g \in \mathbf{G}_i, sm_i(g, \mathbf{P}_c, \mathbf{P}_{si}) = (qv, \mathbf{VR})\}) \ \rangle$$

Informally, we define a semantic content negotiation problem as the process of selecting a pair consisting of an RDF representation $g_{\text{best}}$ from a set of representations available at a semantic web server $sw$ with its validation results set $\mathbf{VR}_{\text{best}}$, in such a way that the selected representation maximises $sw$'s semantic CN measure $sm$ when applying the profile sets $\mathbf{P}_c$ (client preference) and $\mathbf{P}_s$ (server preference).

For example, in our scenario, given the semantic request $sr$, the semantic web server $sw_{\text{yt}}$, at the end of the semantic content negotiation process, it is the RDF representation $R_3$ and is selected given the report set $\{vr_7, vr_8, vr_9, vr_{15}\}$.

In practice, $\langle g_{\text{best}}, \mathbf{VR}_{\text{best}} \rangle$ does not always exist. For example, if $\mathbf{G}_i = \emptyset$. We define $\langle d_{\text{no\_rep}}, \emptyset \rangle$ to indicate that no acceptable response could be found for the given semantic request.

## 5.4 Discussions on S-CON-NEG Applicability

In this section we present parts of the instantiation of our model for enabling flexible and relaxed content negotiation. In this section we follow and build on what was presented in Section 4.3.2 to simulate the use case presented in Section 5.1. We start by modelling the different concepts as classes and interfaces as shown in Figure 5.1. The code is openly available on Github.[9]

The classes `ThirdScenario` and `FourthScenario` simulate the scenarios presented in Section 5.1.1 and Section 5.1.2 respectively. As it follows and reuses parts of `SecondScenario`, we focus on only four parts that answer the following questions:

[9]CN formalisation repository: https://github.com/youctagh/cn-formalisation

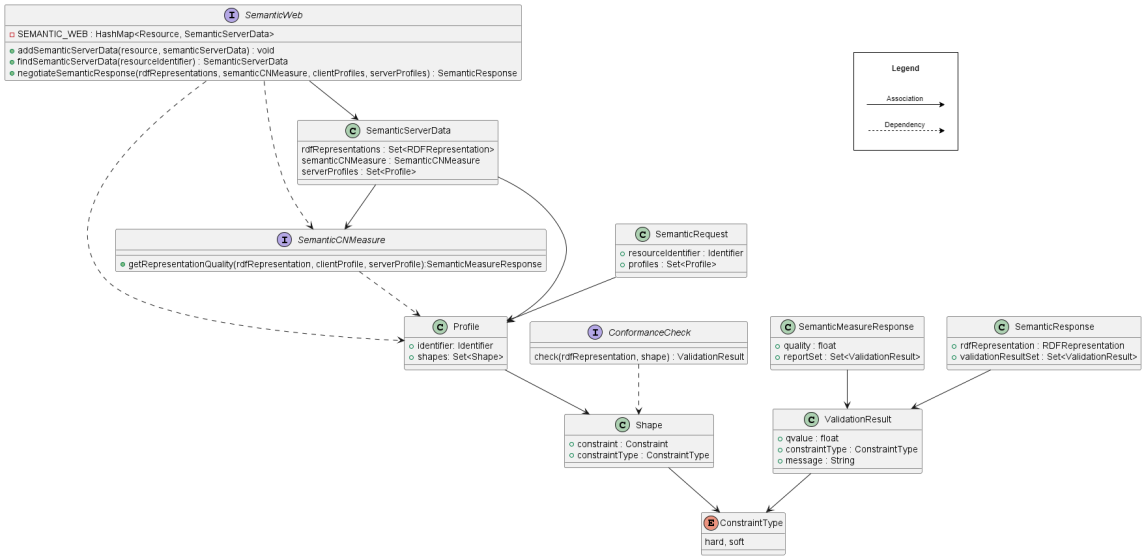



Figure 5.1: A class diagram of our model for flexible and relaxed content negotiation

*How to calculate the acceptability value in on-the-fly and relaxed content negotiation?*
*How to check the conformance of an RDF representation against a profile?*

The snippet in Algorithm 10 defines a semantic CN measure. It uses two utility functions:

**Algorithm 10:** The creation of a semantic CN measure

**input :** An RDF representation $g \in \mathcal{D}_g$, a set of client profiles $\mathbf{P}_c$ and a set of server profiles $\mathbf{P}_s$

**output:** A paire consisting of the final acceptability value in the range $[0, 1]$ of $g$ with the set of its validation results

```
1  float bestClientQuality = 0, clientQuality = 0
2  ValidationResults VR_c_best = ∅, VR_c = ∅
3  foreach p_c ∈ P_c do
4      VR_c = validateRepresentation(g, p_c)
5      bestClientQuality = calculateValidationReportScore(VR_c)
6      if bestClientQuality < clientQuality then
7          bestClientQuality = clientQuality
8          VR_c_best = VR_c
9  float bestServerQuality = 0, serverQuality = 0
10 ValidationResults VR_s_best = ∅, VR_s = ∅
11 foreach p_s ∈ P_s do
12     VR_s = validateRepresentation(g, p_s)
13     bestServerQuality = calculateValidationReportScore(VR_s)
14     if bestServerQuality < serverQuality then
15         bestServerQuality = serverQuality
16         VR_s_best = VR_s
17 return ⟨clientQuality × serverQuality, VR_c ∪ VR_s⟩
```

The first one `validateRepresentation` checks the conformance of an RDF representation to a profile. In our implementation we use *Apache Jena*[10] to manipulate the RDF graphs, perform on-the-fly validation, and generate a report. A snippet of this function is shown in Algorithm 11.

[10]Apache Jena: https://jena.apache.org/

**Algorithm 11:** A function reporting the conformance of an RDF representation to a profile

**input :** An RDF representation $g \in \mathcal{D}_g$ and a profile $p \in \mathcal{P}$
**output:** The validation results when checking conformance of the RDF representation $g$ against the profile $p$

```
1 ValidationResults VR = ∅
2 foreach sh ∈ p do
      // We consider that a cc is known for each shape
3     vr = cc(g, sh)
4     add vr to VR
5 return VR
```

The second one `calculateValidationReportScore` calculates a score given the report generated by the validation. In Scenario 5.1.1, where only on-the-fly validation is required, a scoring function as shown in Listing 12 is used. Note that if there is any violation, the value 0 is returned.

**Algorithm 12:** A scoring function where relax content negotiation is not required

**input :** A validation result set $\mathbf{VR} \subset \mathcal{VR}$
**output:** The acceptability value of the given validation result. Either 0 or 1

```
1 foreach ⟨weight, t, str⟩ ∈ VR do
2     if weight < 1 then
3         return 0
4 return 1
```

In the other case, if we want to relax the negotiation, we use a scoring function like the one shown in Algorithm 13. Note that here we go through the report entries and check whether the violations are of type soft or hard, and act accordingly.

**Algorithm 13:** A scoring function where relax content negotiation is required

**input :** A validation result set $\mathbf{VR} \subset \mathcal{VR}$
**output:** The acceptability value of the given validation result in the range $[0, 1]$

```
1 float qualityValue = 1
2 foreach ⟨weight, t, str⟩ ∈ VR do
3     if t = hard ∧ weight = 0 then
4         return 0
5     else if t = soft then
6         qualityValue = qualityValue × weight
7 return qualityValue
```

## 5.5 Synthesis

The aim of this chapter was to propose a model for flexible and relaxed semantic content negotiation to address Research Question 2. We defined the S-CON-NEG problem model, a specialisation of CON-NEG with relaxed and flexible preferences. First, we presented scenarios motivating the need for both flexible and relaxed negotiation in Section 5.1. We then discussed how such an extension can be achieved, guided by the content negotiation stages in Section 5.2. After that, we presented the model in Section 5.3. Lastly, in Section 5.4, we discussed how it can be practically used in actual implementations.

Our work benefits from the general architectural principles of the Web (cf. Section 2.1.4). First, the principle of *orthogonality* was of great assistance when applied to the content negotiation stages by enhancing content negotiation separately in each stage:

- In *request formulation*, for example, by using SHACL severities to indicate soft and hard constraints.
- In *selection/adaptation*, for example, by proposing new scoring functions.
- In *response indication*, for example, by introducing a new web link relation type.
- In *response interpretation*, for example by extending the semantic interpretation of `application/problem+json` and `Location` header.

- The enhancements made at the *discovery* stage are discussed in Chapter 6.

Second, the *extensibility* principle allowed us to extend content negotiation components as needed, e.g. headers, `Link` relation types, or even selection algorithms. Thirdly, the *error handling* principle was crucial for the selection as well as for the indication of representation conformance and violations via reports. And fourth, the *protocol-based interoperability* principle was valuable, for example by reusing W3C recommendations and IETF technical reports.

In the next chapter we zoom out and re-centre our discussion on the profile dimension and define the finer patterns that shape the negotiation process in this dimension (cf. Research Question 3).

Chapter 6

# Finer Patterns of Content Negotiation in The Profile Dimension

## Contents



In this Chapter, we discuss possible types of finer patterns for handling content negotiation. In particular, we show how some of them can make use of the current web standards and infrastructures, and illustrate them based on the use cases presented in Sections 4.1 and 5.1 in addition to the current efforts of DXWG [161] and IETF [163]. The finer patterns presented here, though illustrated via RDF and SHACL, are general enough to cover negotiation over any structured data model.

To better illustrate the different content negotiation finer patterns, we start by reconstructing and setting up a general use case that will serve as the basis for the following subsections. We then consider several scenarios, each with specific require-

ments. We then propose a solution for each scenario and illustrate the appropriate finer pattern.

## 6.1 Typology of Finer Patterns

### 6.1.1 General Illustrative Use Case

*YoucTagh* maintains a server that provides content about his favourite trees. He uses the URI template `http://youctagh.com/species/{species_name}` to identify each species. For example, the URI `http://youctagh.com/species/abies_numidica` is used to identify *Abies numidica*. For each tree resource, several representations are available including RDF representations, and content negotiation is used to serve the appropriate one given a client's preferences. In these RDF representations, *YoucTagh* claims to be the creator using `dct:creator` and points to the URI that identifies him `<me/>`. This URI can dereference several RDF representations, each using different ontologies: `FOAF`, `DBpedia` and `Schema.org` (see Figure 6.1). And thus, content negotiation in the profile dimension is used.

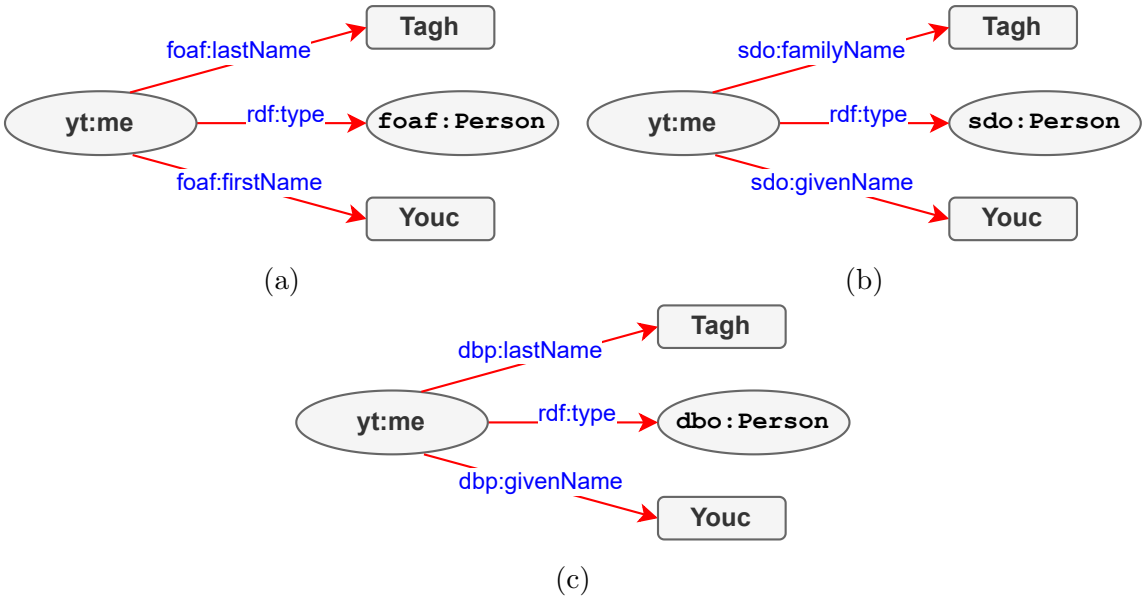



Figure 6.1: RDF graphs describing *YoucTagh* according to 3 ontologies: (a) the FOAF Ontology; (b) the Schema.org ontology; (c) the DBpedia ontology

Depending on the content negotiation pattern, there are different ways for the

client to request the appropriate representation when dereferencing URI <me/>. Similarly, there are different approaches to how the request can be processed. This leads to the various finer patterns discussed in the following sections.

### 6.1.2 Validating Representations with Respect to a Profile

This finer pattern considers that the server can determine whether a representation is conforming to a given profile. Thus, when requesting a resource with a profile, it checks which representation validates the provided profile. This can be done in two ways: either by pre-processing or by checking conformance on the fly. We discuss each method separately in the following. Note that these approaches are not mutually exclusive and can therefore be combined.

**Pre-processed conformance checking:** The server knows a fixed number of profiles that its managed resources may conform to. This way, the server can validate all representations according to each profile in advance, so that it can immediately serve the right representation for a profile without any processing at query time. Preprocessing could be done manually when the representations are added to the server, it can be assumed by a guaranteed data generation process, or it can be validated automatically by relying on a schema language and validator.

In our illustrative scenario, a profile could be well known and impose the use of a certain vocabulary (e.g., the FOAF ontology) or a profile could consist of a SHACL schema such as the one of Listing 4.1.

The advantage of this approach is the reduction in response time, but unforeseen constraints are not handled correctly. In this case, the server can either serve a default representation or respond with an error message.

**On the fly conformance checking:** In this approach, the server is unaware of the concrete constraints that the client is going to send, but assumes that they conform to a specification it implements and are structured in a way it can handle. Unlike the previous approach, in order to select the representation to be returned, the server performs a conformance check of the available representations against the client's constraints. This approach is convenient when processing custom constraints, but at the cost of extra time and resource consumption (e.g. computing power). The SHACL shape of Listing 4.1 is an example of what the client may use as a profile, but arbitrary schemas could be submitted that could adhere to Schema.org or DBpedia or anything else.

**Hybrid conformance checking:** To get the best of both worlds, a hybrid approach to compliance checking can be adopted, where the most commonly requested constraints are pre-processed, and in the selection process, it is first checked whether these constraints have already been processed to avoid a new compliance recheck, otherwise it is computed. An additional caching layer could be used to store newly checked constraints to mitigate time and resource losses if the same constraints are requested again.

### 6.1.3 Profile Compatibility Checking

Similarly to media types, profiles may be organised in hierarchies. That is, representations conforming to a profile $p_1$ may necessarily conform to another profile $p_2$ because the former is more constraining than the latter. Therefore, when a client requests a profile that does not exactly correspond to a fixed list of profiles supported by the server, it could still be possible to serve the client adequately. In some cases, validating each representation individually against the requested profile is not possible or less advantageous. For example, the requested server may not be the one that hosts the representations and may not have access rights on them. In this case, the server may redirect an authorised client to the right origin server.[1] Also, the validation process may be time consuming for some reason (e.g. large representations and complex constraints to validate).

The alternative approach is a profile compatibility checking, where at the server level, instead of holding the representations, the server has the constraints to which the representation conforms (i.e. a documentation). As such, the selection process shifts from: given a set of representations and a set of constraints, select a representation that maximises a score through a validation checking process (i.e. validate each representation against the set of constraints) to: given a set of documentations (each documentation is a set of constraints which validates a representation) and a set of constraints, select a representation that maximises a measure through a containment checking process (i.e. check the containment of the set of constraints with each of the documentations). In a Semantic Web context, one use could be to check the SHACL shape containment of the requested shapes with the one announced by the server [108]. A more general case is profile containment which takes advantage of profiles linked with the `prof:isProfileOf` property in an RDF graph using the profile vocabulary of the DXWG [8, Section 8.3.2]. Similarly to constraint-to-representation, the checking process could be achieved in a pre-processed, on-the-fly

[1] An origin server is the server that holds or can generate a resource representation; other types of servers are the so-called intermediaries (e.g. proxy) [61, Section 3].

or hybrid manner.

In our scenario, *YoucTagh* has a network of institutions he collaborates with that provide data about living organisms to their clients. They restrict access to such data to their customers only but provide formal specifications of the data profiles. So, the server of *YoucTagh* can take requests with arbitrary profiles and redirect to the right origin server of a relevant collaborator relying on the profile compatibility check.

### 6.1.4 Strict vs Flexible Conformance

**Strict conformance:** A profile may define multiple constraints on representations for them to conform. In particular, the more specific features a client demands in its profile preferences, the less a server can satisfy its request. This is because conformance to a profile is typically a binary decision: either a representation *conforms* if it validates all constraints, or *does not conform* if at least one of the constraints is not satisfied. This is what we call *strict* (or *full*) conformance.

To illustrate, a profile defined as a SHACL schema can contain multiple shapes and targets that can be validated individually. In this case, the finer pattern adopts strict conformance when an RDF graph conforms if and only if it validates *all* SHACL constraints.

**Flexible conformance:** A variation of this type is to rely on a conformance measure to serve partially conformant representations. This measure may equal 1 if all constraints are valid, and 0 if all constraints are invalid. This is what we call flexible (partial) conformance: a server may provide a representation that maximises the satisfaction of the clients' preferences, while not fully conforming to the profile. This is to be compared with the notion of hard vs soft constraints discussed in Section 3.4.1. Related, HTTP clients can assign weights to preferences using quality values [61, Section 12.4.2]. Thus, quality values may be considered as a conformance measure by the server.

In our illustrative scenario, a profile may use SHACL with indications of target nodes. A validator can count the number of violations and serve the representation with the least number of them. Additionally, SHACL can assign a severity to constraints [97, Section 2.1.4], which can be used to modulate conformance measure computation.

The disadvantages of this approach are: that it requires on-the-fly checking; that it increases the processing requirements; and that the client may not want to get a partially conformant representation. This can be mitigated using additional headers

or URI query parameters to let the client specify it requires strict handling of its preferences.

### 6.1.5 Unmodified vs Adaptive Negotiation

Another way of satisfying a client's request in a very flexible way is to adapt the response to the client's needs, even if there is no matching representation on the server side. In the case when the server does not have any conformant representation, it is sometimes possible to transform a pre-existing one to accommodate the client. In other cases, it may even be possible to completely generate a representation purely based on the requested profile. A content negotiation pattern that only provides representations that are hosted on the server is qualified as *unmodified* negotiation, while one that provides a modified or generated representation is *adaptive*.

To illustrate how an adaptive architecture can work, we can assume that *YoucTagh* as the sysadmin of his server inspected the logs of client requests over a period of time, and noticed that a non-negligible number of non-compliant requested profiles use *Nature.com* ontology [80]. Although such profiles are not handled by the company, there is some alignment with available profiles as shown in Listing 6.1.

Listing 6.1: An example of the alignment between FOAF and Nature.com ontologies

```
npg:Person rdfs:subClassOf foaf:Person .
npg:familyName rdfs:subPropertyOf foaf:lastName .
npg:givenName rdfs:subPropertyOf foaf:firstName .
```

This procedure has already been used in adaptive content negotiation [114, 110, 115] for example to adapt content for mobile devices. In a Semantic Web context this could take the form of RDF graph repairs as addressed in other work [2, 3, 158]. It could also take other forms (e.g. discarding nodes that violate the constraints, or using alignment techniques to map to a more desired shape). Some additional examples includes: (1) to delete unwanted triples, (2) to use certain units of measurement, (3) to materialise all inferable triples, (4) to adapt an ontology to an OWL profile.

### 6.1.6 Local vs Third Party Mediators

Content negotiation is a layered process with several stages (discovery, request formulation, selection/adaptation, response indication, response interpretation). This feature allows for two options: either to execute the different stages locally, or to externalise them for execution by third parties. This choice depends on the requirements of the use case, such as minimising execution time. In some cases, there can be

intermediaries between the server(s) that host the representation(s) and the server that the client interacts with. In this case, for example, the discovery process may use a third party component to find other representations of the desired resource. The use case in Chapter 7 makes use of such pattern. A third party could be: a mediator that stands between the client and the origin server; an auxiliary service that the server relies on to support content negotiation by profile (e.g. a validation service); an auxiliary service that the client relies on (e.g. an adaptation service). With a third party, it is possible to emulate content negotiation by profile even if the origin server only implements regular content negotiation.

In our running scenario, we have assumed that *YoucTagh* maintains different representations of trees, including an RDF one that uses *YoucTagh*'s own custom vocabulary. As the vocabularies for trees and living species and related data diversify, *YoucTagh* could realise that while he maintains a single RDF representation, he could outsource adaptation to other vocabularies to a service dedicated to keeping alignments between vocabularies up to date, with associated transformation functions. Obviously, the drawback of this approach is the lack of control over the external component, as well as potential latency issues.

### 6.1.7 Discrete vs Non-discrete Representations

A website that only provides representations for a fixed set of URIs can have the complete state of its resources transferred as files organised in folders, even though these may be generated from a monolithic database. This is what we call discrete representations. In other cases, the server can handle an open set of URIs that are composed by the client using query parameters. In this case, one cannot get the complete state of the website as files organised in folders following the path structure of the URIs, because the number of URIs handled by the server is arbitrarily large.[2] This is what we call non-discrete representations. Note that this can be viewed as a content negotiation finer pattern because it allows one to use URI-based content negotiation (as seen in Section 3.4.2) for fine-grained content negotiation by profile. A server having non-discrete representations normally relies on a database management system that group all data together, and should not host the representations per se, except perhaps for caching purposes.

In our running scenario, if we assume *YoucTagh* opted for an external adaptation service, he no longer needs to keep even a single canonical representation for each tree resource. Instead, he can query relevant pieces and post them to the adaptation service that will compose the desired representation. Note that this would only work

[2]Though never infinite, given the restrictions on URI length.

on restricted cases that the adaptation service is able to handle, with well specified inputs and outputs.

The advantage of discrete representation is that it is easy to set up and easy to retrieve. The complete website can be cached if there is not fast-changing data. However, it is not practical in certain use cases, especially when there is a high level of customisability. In a semantic web context, when negotiating in the profile dimension, we can consider both approaches. Some well-identified resources may be served from separate files (that may be "virtual", resulting from a generation script) but an arbitrary set of other representations may be the result of request-specific construction. If the origin server provides a SPARQL endpoint but is not able to do content negotiation by profile, a tool that translates SHACL to SPARQL [46] could be used to achieve content negotiation in this context.

### 6.1.8 Centralised vs Distributed Representations

In traditional content negotiation, different URIs that dereference to different representations can be assumed to denote different resources. If we take this assumption (which is the *unique name assumption*), it is never necessary to request two URIs to obtain the variants of a single resource. However, there is no formal obligation that such assumption holds on the Web, and Semantic Web languages allow several URIs to denote the same resource. For example, to identify the *Abies numidica* species, different servers use different URIs:

- $i_{\text{wd}}$: `<http://www.wikidata.org/entity/Q2004978>`;
- $i_{\text{dbp}}$: `<http://dbpedia.org/resource/Abies_numidica>`;
- or even $i_{\text{yago}}$: `<http://yago-knowledge.org/resource/Abies_numidica>`.

Each of these servers has a subset of the available representations of the resource, meaning that when negotiating with individual servers we are considering only a subset of the potentially valid representations.

Taking into account the potential distribution of representations for a resource leads to patterns for content negotiation that must deal with multiple origin servers in one request. The first step is to axiomatise that, e.g. $i_1$, $i_2$ and $i_3$ actually identify the same resource. This can be done using Semantic Web technologies, such as identity links for example `owl:sameAs` (cf: Section 7.1.3). The sameAs portal [69] provides a service to find equivalence links after extracting them from different sources. The second step is to build a portal that is aware of this distribution and is able to negotiate with the potential servers and aggregate the results. While the distributed

pattern allows the discovery of additional potential representations, it comes at the cost of negotiating with multiple servers and is consequently time consuming.

## 6.2 Synthesis of Finer Pattern Survey

In this section, we provide a summary of the different finer patterns discussed in Section 6.1 with their advantages and disadvantages, all reported in Table 6.1. We additionally provide, for each finer pattern, references to publications or implementations that make use of said type (see the *Reference* column). Such information provides context and exemplifies the different finer patterns.

Further, we consider here the implementations [39, 38, 41, 7] collected in the W3C content negotiation by profile implementation report [162]. They all use the *Unmodified*, *Local* and *Strict* finer patterns types. *Media Types Register* [39] uses a *Non-Discrete* finer pattern (all media types are stored in a `mediatypes.ttl` file and SPARQL queries are used for the extraction) while the others use a *Discrete* finer pattern. *CKAN DCAT Plugin* [41] proposes harvester plugins to allow automatic import of datasets from remote sources, thus we judge that it uses a *Decentralised* finer pattern while the others use a *Centralised* finer pattern. As for the last two finer patterns, all implementations *Pre-process* the validation. However, [38, 39] are both *Profile-compatibility-checking* finer pattern: first they validate the requested profile against the available ones, then they generate the representations with template rendering. Conversely, [7] uses the ProfileWiz tool [9] to generate profiled representations that are used in the content negotiation by profile, making it *Representation-validation* finer pattern. Note that [166, 169] are our own implementations.

| Finer Pattern (FP) | FP Specific | Advantages | Disadvantages | References |
|---|---|---|---|---|
| Representation validation | Pre-processed validation | • Low response time • Cacheable response | • Unable to validate unforeseen constraints | [7, 178, 41] |
| | On the fly validation | • Handle unforeseen constraints | • Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent | [166, 169] |
| | Hybrid approach | • Low response time • Partial cacheable response • Handle unforeseen constraints | • Additional configuration required to accept hybrid approach • Additional resource consumption (e.g. computing power) | [114, 115] |
| Profile compatibility checking | Pre-processed | • Low response time • Cacheable response | • Unable to validate unforeseen constraints | [38, 39] |
| | On the fly | • Handle unforeseen constraints | • Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent | [36] |
| | Hybrid approach | • Low response time • Partial cacheable response • Handle unforeseen constraints | • Additional configuration required to accept hybrid approach • Additional resource consumption (e.g. computing power) | |
| Strict (Full) vs Flexible (Partial) conformance | Strict (Full) | • Low response time • Cacheable response • Compatible with Pre-processed validation, on the fly validation and their hybrid finer FPs | • Only preconfigured default representation is possible • Partial and total conformance of constraints is treated equally • Constraints precedence is not considered | [39, 38, 7, 178, 109, 115, 41] |
| | Flexible (Partial) | • A representation is served even if it partially conforms to the constraints • Constraints precedence is considered | • Only compatible with on-the-fly FP • Additional conformance check time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent | [36, 166, 169] |
| Unmodified vs Adaptive negotiation | Unmodified | • Low response time • Cacheable response | • with unforeseen constraints only error or default response is sent | [36, 39, 38, 7, 178, 41, 166, 169] |
| | Adaptive | • Adapt available representation to unforeseen constraints | • Additional adaptation process time • Additional resource consumption (e.g. computing power) • Caching response is not pertinent | [114, 111, 6, 109, 115] |
| Local vs 3rd party mediators | Local | • Low response time • Full control on the executed process | • Must implement and manage the process locally • More resource consumption (e.g. computing power) | [36, 39, 38, 7, 178, 6, 109, 41, 166] |
| | 3rd party mediators | • The implementation and management of the process is delegated • Less resource consumption (e.g. computing power | • Less control and extends all the risks of the $3^{rd}$ party • Additional time (e.g. transfer and authentication) • Need to Coordinate $3^{rd}$ parties | [115] |
| | Hybrid approach | • The implementation and management of critical processes is done locally • Extends the advantages of (local/$3^{rd}$ party) when used | • Extends the disadvantages of (local/$3^{rd}$ party) when used | [114, 109, 169] |
| Discrete vs Non-discrete Representations | Discrete | • Easy to set up, the default FP for some servers • Easy to retrieve the representation | • Must manage the duplicate representation changes in complex representations | [36, 38, 7, 178, 114, 6, 109, 115, 41, 166, 169] |
| | Non-discrete | • One source of truth for complex representations | • Must be configured and manage the retrieval of representations | [39, 169] |
| Centralised vs Distributed | Centralised | • All representations are available in one place | • Not practical for the Web scale | [36, 39, 38, 7, 166, 178, 6, 109, 115] |
| | Distributed | • Practical for the Web scale • More representations are available | • Less control since the content negotiation is externalised to other servers • Additional time of the distribution request and restitution of response | [41, 169] |

Table 6.1: Content Negotiation Finer Patterns Review

## 6.3 Summary

The aim of this chapter was to explore in more detail finer patterns of content negotiation beyond content negotiation styles, and specifically in the profile dimension, as this is the one of interest in our work, and to address Research Question 3.

We gave a typology of the finer patterns and their practical application in a Semantic Web context, using RDF graphs as representations and SHACL as validation language in Section 6.1. We then synthesised and illustrated these patterns with actual implementations (cf. Section 6.2). We observe that although a large number of finer pattern options are available for content negotiation by profile, only a handful of the possible combinations have been implemented and tested.

To experiment with previously untested combination, and to address Research Question 4, in the next chapter we present our contribution on using identity links to discover and negotiate distributed representations.

# Part III

# Contribution - Experimentation and Validation

Chapter 7

# Identity Links Based Discovery Algorithm

## Contents



In Part II of this dissertation, we focused on the modelling and design aspect of the envisaged semantic content negotiation framework. First, we presented our general model for content negotiation. Then we discussed how we can improve the content negotiation process using semantic validation with the vision of achieving flexible and relaxed interactions. In addition, we explored and discussed finer patterns applicable to content negotiation in the profile dimension.

In this part, we focus on the experimentation and validation of our work. Starting with this chapter, where we discuss our approach to achieve decentralised content

negotiation using Semantic Web technologies. First, in Section 7.1 we examine in more detail a problem arising from the distribution of representations on the Web (Limitation 4). Second, in Section 7.2, we extend CON-NEG to propose a formalisation for the distributed content negotiation problem. Then, in Section 7.3, we present our methodology and how we used identity links such as `owl:sameAs` to discover potential new representations to address the Research Question 4. We present two algorithms to illustrate such use in two content negotiation dimensions. The first is the widely used media type dimension, for negotiating web documents. The second is the profile dimension, for negotiating RDF representations that conform to a profile. After that, in Section 7.4, we first discuss the architecture adopted and the finer pattern combination used. Afterwards, we present the implementation we developed to verify the feasibility of our approach based on the presented algorithms. Finally, in Section 7.5, we present the experiments we have conducted to measure the benefits of our approach to increase the overall availability of resource representations and to assess the time requirements.

## 7.1 The Representation Distribution Problem

The aim of this section is to illustrate the problem of representation distribution (cf. Limitation 4). Since different URIs can be used to identify the same resource, and a server could have different resource representations, representations may be distributed and only a subset of all available representations is considered in a client/server interaction. We can illustrate this with a scenario.

**Story 5.** *YoucTagh* and his friend *Lyncis* are both tree enthusiasts. They each maintain a separate server that provides curated information about their favourite species. Each species could have multiple representations, and therefore content negotiation is used in both servers. While the main purpose is the same, there are several practical differences. *YoucTagh* uses the URI template `<http://youctagh.com/species/{species_name}>` and provides HTML, PDF, plain text and RDF representations. Content is mainly in English, and RDF representations use a custom ontology (cf. Figure 2.1). *Lyncis* uses `<http://lyncis.com/entity/{tree_name}>` as a URI template and provides HTML, XML, JSON in addition to RDF representations. However, the content is mainly in French, and RDF representations use `Umbel` [121] in addition to `Schema.org`. For example, *YoucTagh* identifies the resource *Abies numidica* by the URI `<http://youctagh.com/species/abies_numidica>` and *Lyncis* by the URI `<http://lyncis.com/entity/abies_numidica>`. Similarly, other servers on the Web have their own *Abies numidica* rep-

resentations and identification schemes, e.g. <http://www.wikidata.org/entity/Q2004978> of Wikidata and <http://dbpedia.org/resource/Abies_numidica> of DBpedia. While all these servers are complementary, the challenge is that the representations are siloed.

*YoucTagh* would like to enable its users to know about, find and request these other representations that are spread across multiple servers, especially if none of the representations available on his server is acceptable.

The above situation is possible because of URI aliasing on the Web. However, as a result, it raises questions about identity management in this space, which we examine in Section 7.1.1 as well as Section 7.1.2. We then explore the topic of equivalence links and how they can be useful in Section 7.1.3.

### 7.1.1 Representations and Unique Name Assumption

Open, distributed, accessible and heterogeneous are some of the fundamental characteristics of the Web [42]. This means that in this space, and consequently in the Semantic Web through the RDF data model, anyone can say anything about anything. URIs are used to identify resources, and content providers should not use the same URI to identify different resources to prevent URI collision [88, Section 2.2.1]. However, they are not prohibited from using different URIs to identify the same resource, and these are called URI aliases [88, Section 2.3.1]. Note that the HTTP URI scheme delegates authority over a set of URIs with a common prefix to a particular owner via the IANA URI scheme registry and the Domain Name System (DNS) [88, Section 2.2.2.1]. In this way, different content providers are able to describe the same resource without losing ownership. Figure 7.1 extends Figure 2.3 to illustrate Story 5 and how different representations are served by different servers, although each uses a different URI to identify the same resource.

At the same time, we also know that some of these URIs are subject to content negotiation. As a result, if two URIs identifying the same resource and both of them are subject to content negotiation get dereferenced, we get two sets of representations. Hence, when we negotiate a representation with a content provider, we are only considering a subset of all available alternatives. This is the same as if we were assuming unique names.

The unique name assumption says that different names must refer to different things. If we were to apply this to the Web with what we have seen, it would mean that different URIs must identify different resources, and by contrast all representations of a resource must be served by a single server.

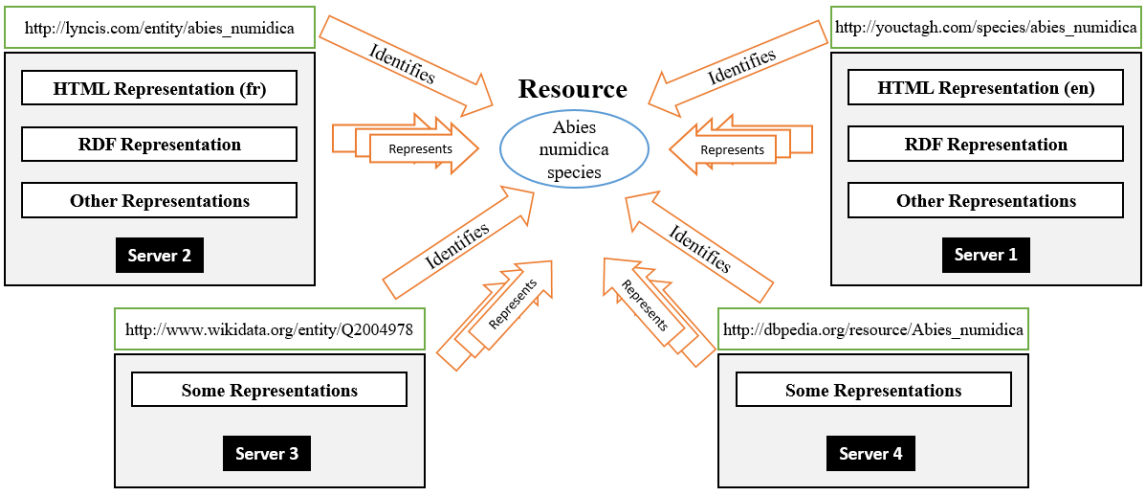



Figure 7.1: A resource identified by different URIs with its distributed representations

In our work we want to negotiate across server boundaries (cf. Limitation 4), and as shown in Figure 7.2, enable content negotiation across the set of representations available on multiple servers.

This may seem difficult to achieve in the Semantic Web, because unlike database semantics, which make unique name assumption as well as closed-world assumption [147, Section 8.2.8], RDF does not. However, OWL solves this by providing the `owl:sameAs` property to assert that two URIs refer to the same thing, as well as `owl:differentFrom` to assert that two URIs refer to different things.

### 7.1.2 Identity Management

The question of whether two entities are actually the same is not new. We can trace it back to *Indiscernibility of Identicals* (a.k.a. Leibniz's law), which relates to *The Ship of Theseus*,[1] that questions identity over time.

As mentioned previously, in database semantics we consider closed-world assumption and unique name assumption, and by using primary keys ambiguity is usually avoided. However, if we consider a scenario where several databases are to be merged, especially in converging domains, the likelihood of overlapping records is high, and we would probably end up with multiple primary keys presumably identifying the

[1]The Ship of Theseus is a paradox regarding identity and change over time: suppose we had a ship that had all its original components replaced over time, would it remain the same object?

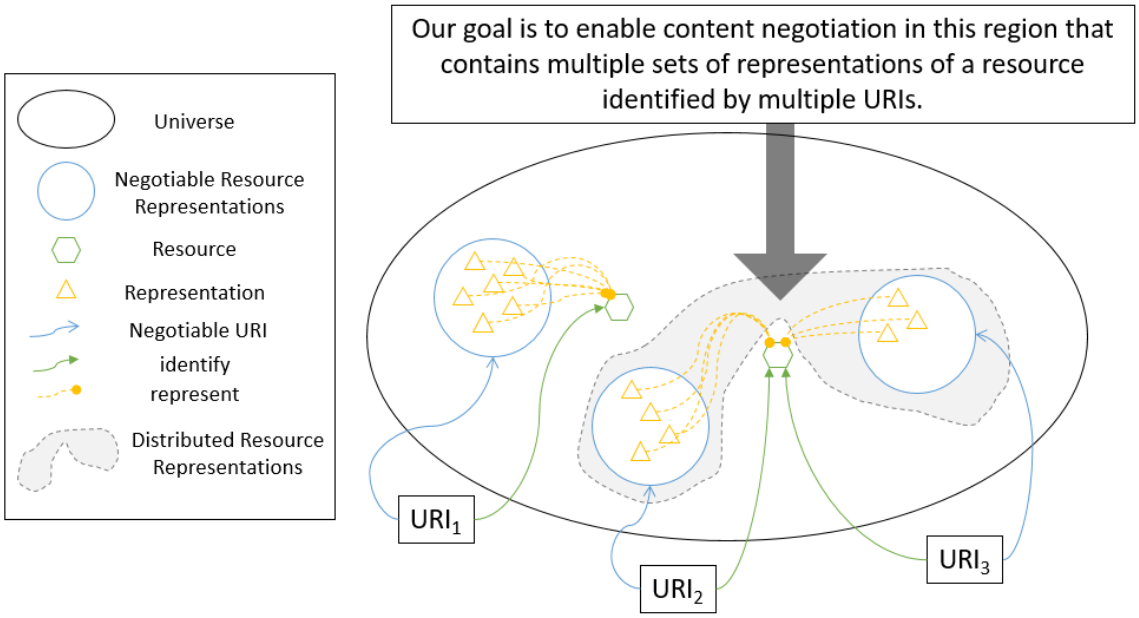



Figure 7.2: Distributed content negotiation problem

same entity. On the Web, and by extension the Semantic Web, the same observations can be made. We would encounter URI aliases (different URIs identifying the same resource), mainly because of the absence of a central naming authority that provides such identifiers. OWL offers `owl:sameAs` to capture the fact that two URIs actually refer to the same entity, e.g. (`<clark_kent>`, `owl:sameAs`, `<superman>`). However, *The Ship of Theseus* question arises again, as the meaning of a URI can change depending on the context in which it is used. An in-depth survey of identity management in the Web of Data has been conducted, which explores this topic in greater detail [141].

In some settings it would be acceptable to use equivalent URIs interchangeably, because precise definition is not paramount, while in other cases one would want a URI that specifically matches its intended meaning. As far as our thesis is concerned, since the main goal of our research is to allow for a more flexible and relaxed content negotiation, we consider such equivalent URIs to be appropriate in this context.

### 7.1.3 Equivalence Links

In Semantic Web a special set of predicates is the set of identity links that assert similarity and equivalence. Their role is to indicate that two distinct identifiers ac-

tually refer to the same entity. A well-known and widely used one is `owl:sameAs`. Other alternative identity links have been proposed in various vocabularies, including (1) `skos:exactMatch` and `skos:closeMatch` from the *SKOS* ontology [122]; (2) `wdt:P2888` from *Wikidata*[2]; (3) `umbel:isLike` in *Upper Mapping and Binding Exchange Layer ontology* [121]; (4) `schema:sameAs` in *Schema.org*; (5) and finally the predicates introduced in the *similarity ontology* [79].

These links could be used in an identity management service to help both users and applications find and discover equivalent identifiers. Several identity management services have been proposed over the years [141, Section 5],[3] such as *RKBexplorer*[4], *sameas.org* [69] *sameas.cc* [12] or also *lodsyndesis* [124].

Equivalence links, in particular `owl:sameAs`, have been addressed in earlier studies where a quantitative analysis of `owl:sameAs` has been undertaken as well as a dataset[5] has been proposed [50].

## 7.2 A Formalisation of the Distributed Content Negotiation Problem

In this section, we consider the scenario described in Story 5 to formalise the problem of content negotiation when representations are distributed across multiple servers (cf. Section 7.1). We reuse CON-NEG and introduce new definitions to accommodate the changes imposed by the new context.

Since a resource can be identified by multiple URIs (URI aliases), we assume that functions to find the different URI aliases could be used to get the identifiers of a given resource.

[2]In Wikidata the property `P2888` is "Exact match": https://www.wikidata.org/wiki/Property:P2888

[3]Note that not all of these services rely on the same identity links.

[4]RKBexplorer: https://rkbexplorer.com/sameas/, the page is no longer accessible. However, archived versions exists, e.g. https://web.archive.org/web/20160909062044/https://rkbexplorer.com/sameas/

[5]The Extended SameAs Network (ESameNet) dataset.

**Definition 7.1: Resource Identifiers Finder**

We define the set of resource identifier finders $\mathcal{RES\_ID}$. A resource identifier finder $res\_id \in \mathcal{RES\_ID}$ is the function that, given a resource $r \in \mathcal{R}$, returns the set of identifiers $\mathbf{I} \subseteq \mathcal{I}$ that identify that resource, formally:

$$\forall r \in \mathcal{R}, res\_id(r) \subseteq \{i \in \mathcal{I} \mid id(i) = r\}$$

For example, if we use a resource identifiers finder $res\_id \in \mathcal{RES\_ID}$ in our scenario, we obtain a set of four IRIs identifying the resource *Abies numidica*:

$$res\_id(Abies\ numidica) = \{\ \texttt{<http://youctagh.com/species/abies\_numidica>}\ ,$$
$$\texttt{<http://lyncis.com/entity/abies\_numidica>}\ ,$$
$$\texttt{<http://www.wikidata.org/entity/Q2004978>}\ ,$$
$$\texttt{<http://dbpedia.org/resource/Abies\_numidica>}\ \}$$

**Definition 7.2: Resource Representations Finder**

We define the function $res\_rep$ that given a resource $r \in \mathcal{R}$ and a client constraints $\mathbf{C}_c \subset \mathcal{C}$ finds the acceptable representations of that resource on the Web $\mathcal{W}$, formally: Let $res\_id \in \mathcal{RES\_ID}$ be a resource identifiers finder.

$$res\_rep(r) = \{d \in \mathcal{D} \mid \exists i \in res\_id(r) \wedge \exists \langle \mathbf{D}_i, m_i, \mathbf{C}_i \rangle \in w(i)$$
$$\wedge\ d \in \mathbf{D}_i \wedge m_i(d, \mathbf{C}_c, \mathbf{C}_i) > 0\}$$

In this context, each of the servers has its own CN measure and is unaware of the other servers' data. We assume that the web agent receiving the set of representations (this could be the client or a third party mediator depending on the finer pattern used) has a function that selects a representation from a set of representations that is considered to be the best one.

**Definition 7.3: Best Representation Selector**

We define the set of best representation selectors $\mathcal{BEST}$. A best representation selector $best \in \mathcal{BEST}$ is a function that selects a representation from a set of representations that it deems best, formally:

$$best : 2^{\mathcal{D}} \rightarrow \mathcal{D}$$

With *best* defined only on non-empty sets. It also satisfies the following two properties:

$$\forall\, \mathbf{D} \subseteq \mathcal{D},\ best(\mathbf{D}) \in \mathbf{D}$$

$$\forall\, \mathbf{D}_1, \mathbf{D}_2 \subseteq \mathcal{D},\ \mathbf{D}_1 \subseteq \mathbf{D}_2,\ best(\mathbf{D}_2) \in \mathbf{D}_1 \implies best(\mathbf{D}_1) = best(\mathbf{D}_2)$$

Similarly, we assume that the web agent receiving a set of pairs of RDF representations and their validation results has a function that selects one pair from this set which is considered to be the best.

**Definition 7.4: Best Semantic Representation Selector**

We define the set of best semantic representation selectors $\mathcal{SBEST}$. A best semantic representation selector $sbest \in \mathcal{SBEST}$ is a function that selects a pair from a set of pairs consisting of an RDF representation with its validation results that it deems best, formally:

$$sbest : 2^{\mathcal{D}_g \times 2^{\mathcal{VR}}} \to \mathcal{D}_g \times 2^{\mathcal{VR}}$$

Similarly, *sbest* is defined only on non-empty sets. It also satisfies the following two properties:

$$\forall\, \mathsf{E} \subseteq \mathcal{D}_g \times 2^{\mathcal{VR}},\ sbest(\mathsf{E}) \in \mathsf{E}$$

$$\forall\, \mathsf{E}_1, \mathsf{E}_2 \subseteq \mathcal{D}_g \times 2^{\mathcal{VR}},\ \mathsf{E}_1 \subseteq \mathsf{E}_2,\ sbest(\mathsf{E}_2) \in \mathsf{E}_1 \implies sbest(\mathsf{E}_1) = sbest(\mathsf{E}_2)$$

## 7.3 Employ Content Negotiation in a Decentralised Semantic Context

On the Web, representations of a resource may be scattered across different servers. As a consequence, a client making a request to one of these servers, and negotiating for a representation from among those it holds, is only considering a subset of all the variants available on the Web as a whole.

Fortunately, as discussed in Section 7.1.3, identity links such as `owl:sameAs` are used to indicate that two URIs actually identify the same thing (in our use case, the same species). In this section we propose an approach that uses these identity links to discover new resource identifiers, and by implication, additional potentially acceptable representations.

In the next two sections, we present two algorithms that demonstrate how such an approach can be applied. We start with Algorithm 14, which presents the idea in the context of the well-known media type dimension. Then Algorithm 15 illustrates how such an improvement could also be beneficial in a semantic context, by negotiating RDF graphs that conform to a profile. Finally, we discuss how the approach could be generalised to other dimensions and adapted to flexible and relaxed content negotiation.

### 7.3.1 Basic Content Negotiation Context

The negotiation on the server side starts when a client makes a request for a resource, which consists of an identifier $i_q$ and a set of constraints $\mathbf{C}_q$ (constraints to define some acceptable media types). We consider that the server uses a function $res\_id$ to find the identifiers of some resource on the Web. The client expects to receive in response a representation $d_{\text{best}}$ that validates $\mathbf{C}_q$, and possibly a set of possible alternative identifiers $\mathbf{I}_a$ to continue the negotiation if desired.

The server managing the identifier $i_q$ checks whether any of the representations it has validates $\mathbf{C}_q$. If not, a common practice is to respond with a `406 Not Acceptable` or `404 Not Found` error or with a preconfigured default (even invalid) representation [61, Section 12]. In our approach, in the negative case, the negotiation continues by first looking for a set of alternative identifiers that identify the same resource $\mathbf{I}_{\text{s}}$.

A straightforward implementation could be to redirect the client to these other potential representations by sending a `300 Multiple Choices` response to indicate other alternatives that could possibly satisfy the constraints.[6] Alternatively, the server can go a step further to check whether the constraints are satisfied by one of these identifiers as described in Algorithm 14. After obtaining the set of identifiers $\mathbf{I}_{\text{s}}$, the server iterates over it and checks whether there is a representation $d_{\text{s}}$ that satisfies the constraints $\mathbf{C}_q$. In this proposal, the server responds with the first such $d_{\text{s}}$ found. For the rest of the identifiers, if a valid representation could be found, the server stores its identifier $i_{\text{s}}$ in the set of alternative IRIs $\mathbf{I}_a$ in order to form the final answer pair $\langle d_{\text{best}}, \mathbf{I}_a \rangle$. If no representation could be found, only then the server responds with an error, in our case $\langle d_{\text{no\_rep}}, \emptyset \rangle$.

The $res\_id$ function is used to find a set of alternative identifiers for the requested resource. There are several ways of doing this. For example, (1) a server may have a local identity management system;[7] (2) if an RDF representation of the resource

[6] In this case the negotiation style is the *reactive* one.

[7] Jaffri proposed an example of an identity management system with the function

**Algorithm 14:** Decentralised CN pseudocode using CON-NEG

**input :** A request $q \in \mathcal{Q}$ for a resource $r \in \mathcal{R}$ where $q = \langle i_q, \mathbf{C}_q \rangle$ and a resource identifier finder $res\_id \in \mathcal{RES\_ID}$

**output:** A pair consisting of a representation $d_{\text{best}}$ that validates $\mathbf{C}_q$ if available, and potentially a set of possible alternative identifiers $\mathbf{I}_a$

```
1  I_a = ∅
2  d_best = negotiate(q)
3  if d_best ≠ d_no_rep then
4      return ⟨d_best, ∅⟩
5  I_s = res_id(r)
6  foreach Identifier i_s of I_s do
7      d_s = negotiate(⟨i_s, C_q⟩)
8      if d_s ≠ d_no_rep then
9          if d_best = d_no_rep then
10             d_best = d_s
11         else
12             add i_s to I_a
13 if d_best ≠ d_no_rep then
14     return ⟨d_best, I_a⟩
15 else
16     return ⟨d_no_rep, ∅⟩
```

exists, extract alternatives from the graph on the fly; (3) rely on a third party service to provide such alternative identifiers, e.g. `http://sameas.org`; (4) or rely on a remote identity management service, e.g. `http://sameas.cc`.

In practice, the `negotiate` method makes a `GET` request with the received preferences. This is an example where a server becomes a client [61, Section 3.3]. This algorithm can be used by a mediator (a third party) who has no representations such as the one presented later in Section 7.4. However, if the algorithm is used by an origin server, the first call for `negotiate` (line 2) checks for local representation that validates the preferences.

### 7.3.2 Semantic Context Content Negotiation

The same idea could be carried over from the Web of documents to the Web of data, but with some modifications to adapt it to the Semantic Web context. First, the server assumes that the client is negotiating an RDF representation with its validation results, and therefore constraints need to be defined for this type of representations (e.g. using SHACL [97] or ShEx [137]). Second, the server can use the First Found First Served approach as illustrated above (i.e. when iterating over the alternative identifiers $\mathbf{I}_a$, it serves the first RDF representation that validates the requested profiles $\mathbf{P}_q$, with its validation results).

Algorithm 15 illustrates a more generic serving method, where the server has two variables $\langle g_{\text{best}}, \mathbf{VR}_{\text{best}} \rangle$ and $i_{\text{best}}$ to store the best RDF representation with its validation results and the identifier of the best RDF representation respectively. The function *sbest* is used to selects the best pair of RDF representation and its validation results considering a new valid RDF representation and the current best RDF representation with their respective validation results. As discussed previously, the decision could be based on a scoring function (cf. Table 5.1).

These algorithms could be further generalised to other dimensions such as *language* (i.e. to negotiate a representation that uses a preferred language) by following the same model and creating appropriate scoring functions.

In practice, when implementing the `negotiate` method, we have two cases: (1) we assume that the other servers can also handle content negotiation in the profile dimension. Then we use for example the `accept-profile` header to convey the constraints as we received them; (2) we assume that the other servers cannot handle content negotiation in the profile dimension. In this case, we request any RDF representation and then take on the burden of validation and conformance checking. The second assumption is the one made in our proposal (cf. Section 7.4).

---

`findEquivalence` [89]

**Algorithm 15:** Decentralised CN pseudocode using S-CON-NEG

**input :** A semantic request $sr \in \mathcal{SR}$ for a resource $r \in \mathcal{R}$ where $sr = \langle i_q, \mathbf{P}_q \rangle$, a resource identifier finder $res\_id \in \mathcal{RES\_ID}$ and a best representation selector $sbest \in \mathcal{SBEST}$

**output:** A triple consisting of an RDF representation $g_{\text{best}}$ that validates $\mathbf{P}_q$ if available with its validation result set $\mathbf{VR}_{\text{best}} \subset \mathcal{VR}$ and potentially a set of possible alternative identifiers $\mathbf{I}_a$

$\mathbf{I}_a = \emptyset$
$\langle g_{\text{best}}, \mathbf{VR}_{\text{best}} \rangle = \texttt{negotiate}(sr)$
**if** $\langle g_{best}, \boldsymbol{VR}_{best} \rangle \neq \langle d_{no\_rep}, \emptyset \rangle$ **then**
    **return** $\langle g_{best}, \boldsymbol{VR}_{best}, \emptyset \rangle$
$\mathbf{I}_{\text{s}} = res\_id(r)$
$i_{\text{best}} =$ <>
**foreach** *Identifier* $i_s$ *of* $\boldsymbol{I}_s$ **do**
    $\langle g_{\text{s}}, \mathbf{VR}_{\text{s}} \rangle = \texttt{negotiate}(\langle i_{\text{s}}, \mathbf{P}_q \rangle)$
    **if** $\langle g_s, \boldsymbol{VR}_s \rangle \neq \langle d_{no\_rep}, \emptyset \rangle$ **then**
        **if** $sbest(\{\langle g_s, \boldsymbol{VR}_s \rangle, \langle g_{best}, \boldsymbol{VR}_{best} \rangle\}) = \langle g_s, \boldsymbol{VR}_s \rangle$ **then**
            $\langle g_{\text{best}}, \mathbf{VR}_{\text{best}} \rangle = \langle g_{\text{s}}, \mathbf{VR}_{\text{s}} \rangle$
            **if** $i_{best} \neq$ <> **then**
                add $i_{\text{best}}$ to $\mathbf{I}_a$
            $i_{\text{best}} = i_{\text{s}}$
        **else**
            add $i_{\text{s}}$ to $\mathbf{I}_a$
**if** $\langle g_{best}, \boldsymbol{VR}_{best} \rangle \neq \langle d_{no\_rep}, \emptyset \rangle$ **then**
    **return** $\langle g_{best}, \boldsymbol{VR}_{best}, \boldsymbol{I}_a \rangle$
**else**
    **return** $\langle d_{no\_rep}, \emptyset, \emptyset \rangle$

Similarly, this algorithm can be adapted to achieve flexible and relaxed content negotiation. The *sbest* function would be based on the validation results, taking into account hard and soft constraints.

Finally, as some resources have a high number of alternative identifiers (cf. Figure 7.6), it may be necessary to have a halting mechanism. Some options are: (1) consider only a subset of identifiers; (2) a duration threshold after which, respond with the current best option.

## 7.4 Practical Usage of Decentralised Content Negotiation

The aim of this section is twofold: on the one hand, we test an unprecedented combination of content negotiation finer patterns in Section 7.4.1. On the other hand, we assess the feasibility and applicability of our approach by proposing a third party mediator that a client can use to negotiate for semantic content on servers even if they are not semantically enabled, meaning that they cannot perform content negotiation in the profile dimension (cf. Section 7.4.1).

### 7.4.1 Architecture Choice

In this section we present the architecture used to create a mediator (a third party negotiator) based on the algorithms presented earlier. The applied combination of content negotiation finer patterns differs from the existing approaches already available, as can be seen in Table 6.1 (cf. reference [169]).

In more detail, the architecture addresses the following scenario: a mediator provides the ability to negotiate for web content in two dimensions, media type and profile. A URI scheme relative to the mediator's URI is used to identify any resource on the Web. The mediator does not host any data itself, but fetches it dynamically from the Web at request time. To identify which servers to contact, it relies on a third-party service *sameAs.org*. This service has an API that takes a URI as input and returns a list of URIs that the service considers to identify the same resource. The URIs in the list are then sequentially requested for validation against the client's constraints. The general architecture in the media type dimension is shown in Figure 7.3.

Upon receiving a request from a client with the initial identifier and constraints (see. Step 1), the portal sends a request to the origin server, i.e. the server that manages the URI specified by the client with the requested constraints (see. Step 2).

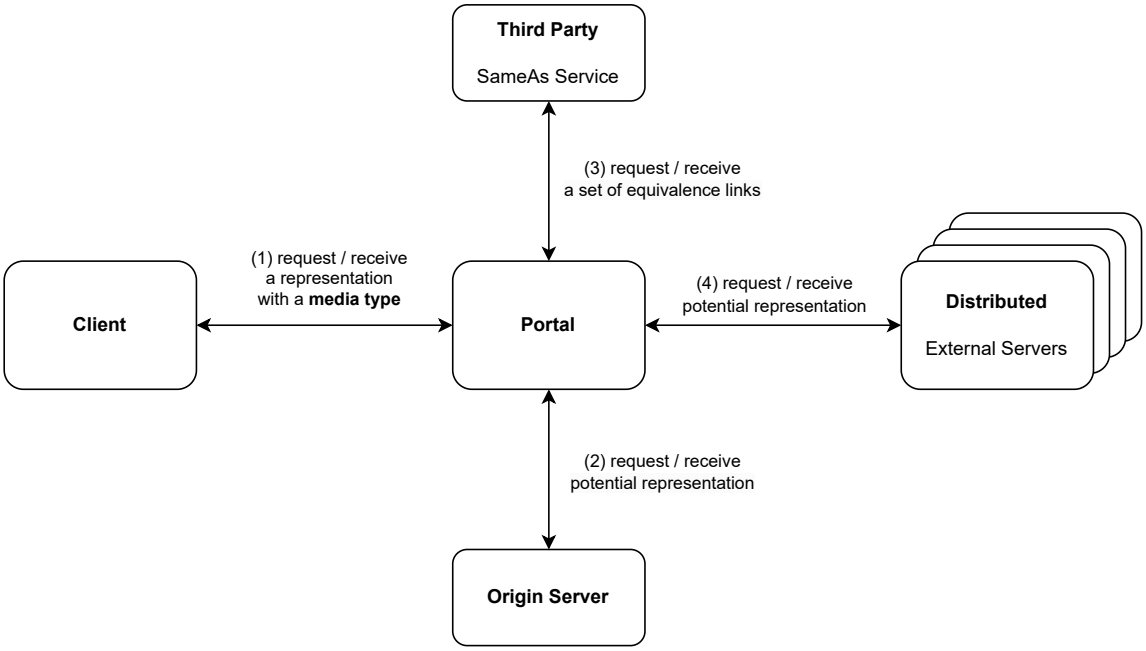



Figure 7.3: Overview of the decentralised content negotiation architecture in the media type dimension

If no acceptable representation is found, the portal requests a set of alternative identity links from a third party service (see. Step 3). It repeats Step 2, but this time with the servers that manage the URIs received from the identity management system (Step 4), and then respond accordingly.

A similar architecture is used for the profile dimension, as shown in Figure 7.4. However, since not all servers can achieve content negotiation in this dimension, in Steps 2 and 4 the portal requests any RDF representation and perform validation on the fly on the portal to check that the representation conforms to the requested profile.

If we analyse our proposal for the profile dimension to extract the finer patterns used, first, we can see that it falls into the *representation-validation* pattern with *on-the-fly* validation. Note that there is no caching enabled in the current version, if such a caching mechanism existed it would fall into the *hybrid validation* pattern. It does not adapt the RDF representations if they do not conform to the requested profile, which makes it fall into the *unmodified* pattern. Since the alternative representations are found through the third party sameAs portal and the validation is done locally, this makes it *hybrid* in this pattern type. At the same time, since the

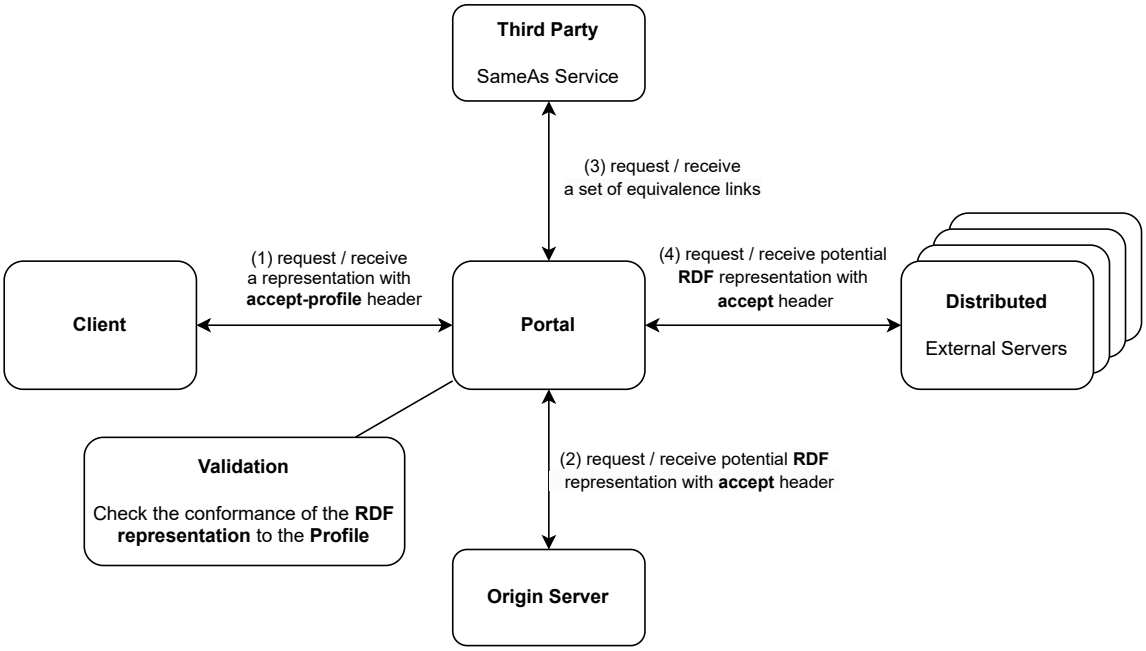



Figure 7.4: Overview of the decentralised content negotiation architecture in the profile dimension

representations are not on a central server, this makes the suggestion *distributed*. In such a case, no prior assumption could be made about how the representations are stored (i.e. either *discrete* or *non-discrete* patterns could be used). Finally, the proposal takes into account the severity of the constraints, either in the validation or in `isABetterRepresentationThan`, to select the best representation to serve, thus making it conform to the *flexible conformance* pattern.

### 7.4.2 Mediator Implementation

We provide a Java implementation of our algorithms. Our prototype[8] is built using the *Spring framework*.[9] We provide two endpoints `</dcn/api/media-type>` and `</dcn/api/profile>`. The resource URI is provided using the *iri* query parameter, and constraints are passed in a header based manner, `accept` and `accept-profile` for the media type and profile dimensions respectively. The `Alternates` header is

[8]Github repository: https://github.com/YoucTagh/decentralised-cn
[9]Spring Framework: https://spring.io/

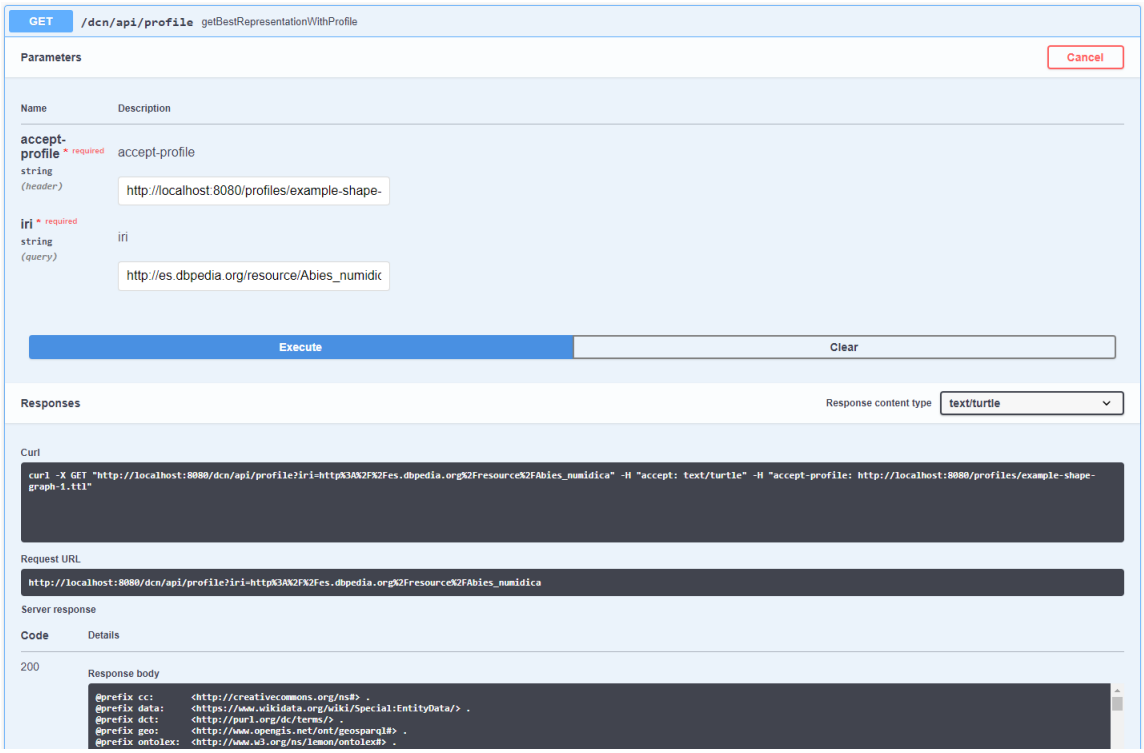



Figure 7.5: A request sent using the Swagger-UI to get a representation that conforms to a profile

used to provide the set of alternative URIs.[10] The *sameAs service* [69] is used to obtain the set of alternative URIs. *SHACL* is used to write the constraints that RDF representations must conform to, while *Apache Jena*[11] is used to manipulate the RDF graphs and perform validation. Swagger[12] is used to document the API.

Figure 7.5 shows a request example where a client wants to obtain a representation of the resource *Abies numidica* identified by the URI `<http://es.dbpedia.org/resource/Abies_numidica>` and conforming to a SHACL shape graph. Note that the origin server cannot provide an RDF representation conforming to that profile. The same request could be issued using curl as shown in Listing 7.1.

Listing 7.1: The same request sent in Figure 7.5 but using curl

```
curl -v http://localhost:8080/dcn/api/profile?
```

[10] The same approach is also used by *dbpedia* to provide alternatives
[11] Apache Jena: https://jena.apache.org/
[12] Swagger UI: https://swagger.io/tools/swagger-ui/

```
iri=http://es.dbpedia.org/resource/Abies_numidica
-H "accept-profile:
http://localhost:8080/profiles/example-shape-graph-1.ttl"
```

## 7.5 Experiments

After having tested the proposed implementation in Section 7.4.2 with a few manually selected URIs, we carried out experiments in order to test the following hypotheses: (1) In case no acceptable representation is available at the initial URI (when negotiating some RDF serialisation, such as `text/turtle`, `application/rdf+xml` ...), our approach would increase the chance of finding an alternative using identity links when available. (2) The *sameAs.org* portal could be used as a reliable third party medium to find similar entities (e.g. it does not have an API call limit). This section provides details on the selected dataset, the experimental setup, and an analysis of the results obtained. The code, data and results of the experiments are publicly available.[13]

**Data** Our data collection methodology is as follows: (1) collect 5+ sets, each with 75+ URIs. Each set of URIs has a homogeneous number of identity links. (2) Test if the *sameAs.org* service can handle and allow a large number of API calls. In addition, we want context-free entities, meaning that no assumptions are made about the type of data. To this end, we use the Wikidata identifiers for items.[14] This means that requests with URIs following the template `http://www.wikidata.org/entity/Q{id}` are sent to the *sameAs.org* API by replacing the `{id}` part in the URI with an integer. In these experiments, the identifiers range from 1 to 5000. The statistics of the responses are provided in Figure 7.6.

From all the responses we create six subsets: $[1, 5]$, $[10, 15]$, $[25, 30]$, $[45, 50]$, $[70, 75]$, $[+100]$, each with 100 URIs. A subset $[x, y]$ means that all its URI elements have between $x$ and $y$ identity links. URIs are added to the relevant subset until it is full (i.e. first found first added).

**Experimental Setup** The code experiments are written in Java and the Apache Jena framework is used to manipulate RDF graphs and perform validation of SHACL shape graphs. The used machine has an Intel(R) Xeon(R) CPU E3-1505M v6 @3.00GHz processor with 16GB of RAM. For each of the six subsets created:

[13] Github repository: https://github.com/YoucTagh/decentralised-cn-experiment
[14] Wikidata identifiers: https://www.wikidata.org/wiki/Wikidata:Identifiers

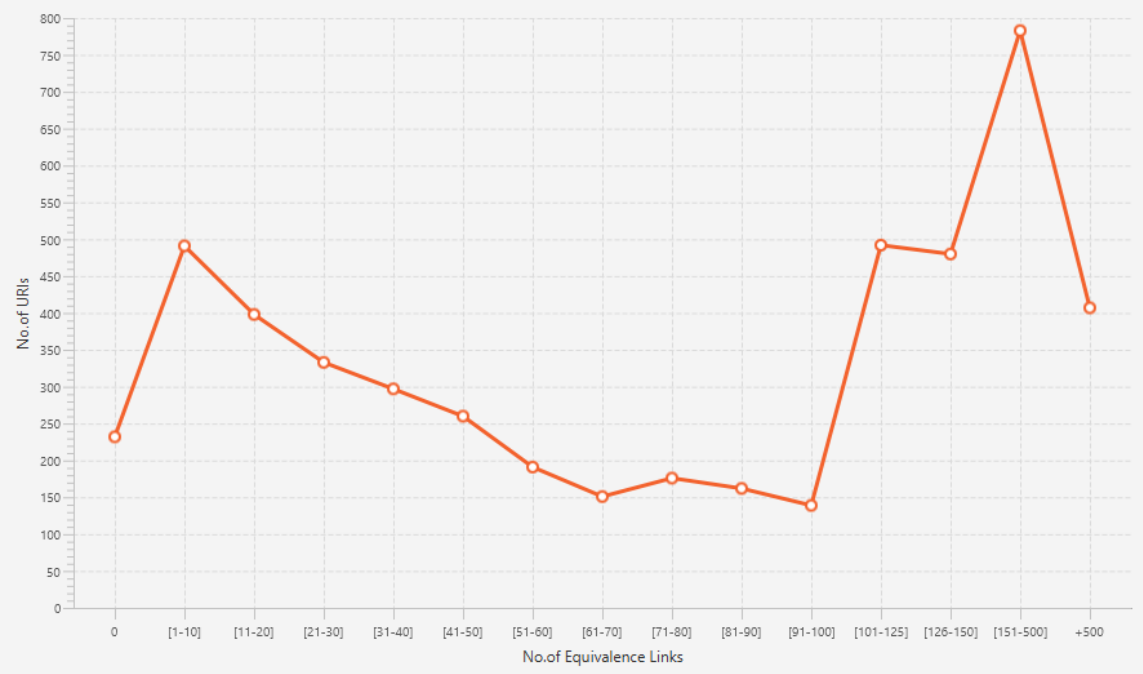



Figure 7.6: The statistics of the responses

1. start by deleting the wikidata URI from the identity links[15]
2. randomly choose a URI from the identity links to be our initial URI[16]
3. request a representation that validates the constraints:
   (a) **HTML:** request an HTML representation.
   (b) **RDF:** request one of the following RDF representations (*Turtle, RDF/XML, N3*).
   (c) **Turtle + SHACL:** request an RDF representation with a Turtle serialisation that validates a SHACL shape graph.[17]

If a representation is found, increment the *Initial URI* score. Otherwise, test whether any of the identity links validate the constraints. If so, increment the *SameAs URI*

[15]The *sameAs.org* service sends back the identity links, if any, plus the requested URI.
[16]A seed is used in the random selection to allow reproducibility of the results.
[17]The SHACL constraints express that the representation with `rdfs:label` and `rdfs:comment` should have at least one language tag in French or English (Available in the aforementioned Github repository).

Table 7.1: Decentralised Content Negotiation Experiment Results

| Constraint | Metric | [1,5] | [10,15] | [25,30] | [45,50] | [70,75] | [+100] |
|---|---|---|---|---|---|---|---|
| HTML | Initial URI | 5 | 20 | 22 | 31 | 20 | 17 |
| | SameAs URI | 13 | 79 | 78 | 69 | 80 | 83 |
| | Avg Time (ms) | 863 | 944 | 845 | 871 | 1077 | 822 |
| RDF | Initial URI | 53 | 44 | 37 | 40 | 29 | 25 |
| | SameAs URI | 47 | 56 | 63 | 60 | 71 | 75 |
| | Avg Time (ms) | 310 | 381 | 657 | 480 | 556 | 393 |
| Turtle + SHACL | Initial URI | 15 | 19 | 15 | 20 | 13 | 11 |
| | SameAs URI | 7 | 17 | 13 | 8 | 8 | 8 |
| | Avg Time (ms) | 277 | 344 | 410 | 400 | 460 | 317 |

score. For each of the subsets and each constraint type, the average response time is calculated.[18]

**Results** The results of the experiments are depicted in Table 7.1. They show that, at best, only 30% of the randomly selected URIs have a representation that validates the *HTML* constraint, while up to 50% validate the *RDF* constraint (we believe that this augmentation is rational, since the new URIs are found through identity links that are expressed in RDF) and, understandably, at most 20% validate the *Turtle + SHACL* constraint. This low percentage is due to either a broken URI or a non-conforming constraint. For each of the subsets we see a non-zero value of the *SameAs URI* score, ranging from 69% to 83% for the *HTML* constraint in all subsets except the first with only 13% this is partly due to the low number of identity links only $[1, 5]$. For the *RDF* constraint, the contribution is $56 - 75\%$, and for the *Turtle + SHACL* constraint, the addition is noteworthy at $7 - 17\%$.

**Experiment Reproducibility** To reproduce the experiment, a main command line interface is provided, when launched a menu with the available options is displayed. The first option will start the data collection process. The second option will use the collected data to start the experiment process. The third option is available to produce a human readable version of the results.

[18]For constraint (1) and (2), requests are sent using the HTTP `HEAD` method.

## 7.6 Summary

In this chapter we proposed an approach to achieve decentralised content negotiation using identity links (cf. Research Question 4). First, we discussed the problem of representation distribution. We showed that since different URIs may be used to identify the same resource, and different servers may have different resource representations, this leads to a scenario where representations are distributed. As a consequence when engaging in content negotiation, only a subset of all available representations is considered. We then looked at related topics, some of which contribute to the problem, such as unique name assumption, and some of which we would consider when building our solution, such as identity links (cf. Section 7.1). Second, we extend CON-NEG to propose a formalisation for the distributed content negotiation problem (cf. Section 7.2). Then, we presented the methodology and how we used identity links such as `owl:sameAs` to discover potentially acceptable representations (cf. Section 7.3). In that section, we introduced two algorithms for negotiating web content in two dimensions, the first being the media type for negotiating web documents, and the second being the profile for negotiating RDF representations that conform to a profile. After that, we presented a third party negotiator based on the proposed algorithms, which enabled the testing of a finer pattern combination previously unseen. Finally, we performed an evaluation of our approach, and the results showed that it contributes to an overall increase in the availability of resource representations (cf. Section 7.3).

This chapter complements Chapter 5 by proposing our contribution to improve content negotiation at the *discovery* stage by using identity links to discover a new subset of representations available on the Web but not identified by a unique URI.

Chapter 8

# Towards Facilitating Heterogeneous Systems Interaction

## Contents



In *The Semantic Web* [21], Tim Berners-Lee and colleagues presented a scenario where agents interact with each other to achieve some goals. For example, agents may need to extract knowledge from some servers. If multiple representations are present, agents may need to engage in content negotiation, which can prove difficult because agents do not always have a priori knowledge of the content negotiation properties used by a web server, e.g. the content negotiation styles used in negotiation, or the preferred means to convey constraints (cf. Limitation 5).

To this end, in this chapter we discuss our contribution to facilitate the interaction of heterogeneous systems when content negotiation is required. First, in Section 8.1, we provide an overview of concepts from MAS research that would be relevant to enabling semantic content negotiation in such systems (cf. Limitation 5). Second, in Section 8.2, we describe a use case for Web-based MAS where content negotiation is required and semantic content negotiation is valuable. Then, in Section 8.3, we present the content negotiation ontology that enables us to capture knowledge about

the content negotiation characteristics of web servers. After that, in Section 8.4, we illustrate the steps needed to bring about and design such a system. Finally, in Section 8.5, we discuss our practical implementation to illustrate how the use case requirements are met.

## 8.1 Content Negotiation in Multi-Agent Systems

In the previous sections, we have focused our discussion on the client-server architecture where the end user is typically a human. This means that we rely on his cognitive abilities to make the right request (e.g. `GET`) to the right server with the right preferences (e.g. `PNG` image representation otherwise a `JPEG`), conveyed with the right means (e.g. `accept` header with q-values), and react to the response accordingly (e.g. in case of a `406 Not Acceptable` status code). However, not only human agents use the Web, as discussed earlier (cf. Section 2.2), the Semantic Web aims to enable software agents to consume and use content in a meaningful way. Thus, such software agents would be intelligent and would need to interact with each other and exhibit some intelligent negotiation capabilities. Therefore, we believe that research from MAS would be of great help in enabling semantic content negotiation in such a heterogeneous space. MAS is a vast domain, and we will explore the facets of interest to us in order to tackle the problem of semantic content negotiation of autonomous agents. In Section 8.1.1 we start with a general discussion on MAS and autonomous agents. In Section 8.1.2, we discuss the presence of MAS on the Web, and specifically how the use of semantic content negotiation would improve the current state of affairs in this area.

### 8.1.1 Multi-Agent Systems

In our work we use the term *agent* to denote a hardware or software-based computer system that is: First, *autonomous*, i.e., no direct human intervention is required and it has control over its actions and internal state. Second, *social* i.e. interacting with other agents. Third, *reactive*, i.e. it perceives its environment and reacts accordingly to changes. Fourth, *proactive* i.e. able to take the initiative in some behaviour and not just react to changes in the environment [184, Section 1.a].

A MAS consists of agents and their *environment*, where such an environment could be for example the physical world or the Internet [184, Section 1.a], or in our case the Web i.e. Web-based MAS. More precisely, we consider a new generation of autonomous systems on the Web called *hypermedia MAS* [43]. Here, the environment is considered as a first-class abstraction [181], providing agents with functionalities

such as: providing access to the external environment (e.g. Internet of things (IoT) devices); offering an abstraction layer for modelling, representing and programming non-autonomous entities; facilitating interaction and communication between agents.

### 8.1.2 Web-based Multi-Agent Systems

The vision of a hypermedia MAS represents all autonomous and non-autonomous entities in the environment as Web resources. A Web resource is anything that can be identified by an IRI, and hyperlinks can be used to interrelate them in a way that supports discoverability. As such, agents are able to navigate the hypermedia themselves to discover resources in the MAS [43, Section 3.1]. However, in such a heterogeneous space, where different agents (both human and software) with different abilities/preferences would be present, and at the scale of the Web, customised representations and support for adaptive knowledge exchange would be critical [65]. However, native mechanisms such as content negotiation are underutilised (Limitation 5). For example, an agent could create a profile and rely on profile-based negotiation to request representations that it can process. Such semantic content negotiation would be very advantageous in systems where the following properties apply:

- clients may have diverse preferences for variants;
- servers cannot have a priori knowledge of the clients' preferences and abilities;
- clients cannot have a priori knowledge of variants;
- and effective negotiation of variants has a strong impact on system efficiency.

In this chapter, we address this very limitation and describe our proposal for facilitating the interaction of heterogeneous systems and providing software agents with the ability to autonomously engage in content negotiation.

## 8.2 Use Case for Content Negotiation in Web-based MAS

In this scenario, we consider a person *Bob* who has an assistant software agent called *Bobajob* that manages *Bob*'s smart home. For example, if *Bob* has an enquiry about a particular topic, *Bobajob* will search the Web for an answer that matches *Bob*'s preferences, aggregate the results, and display them on *Bob*'s smart TV.

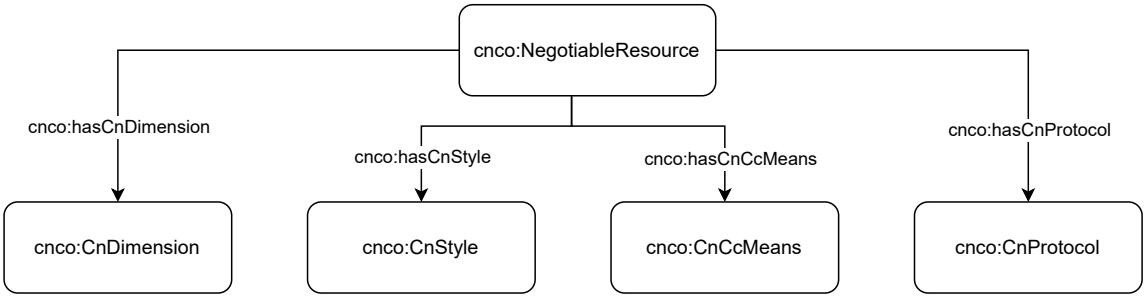



Figure 8.1: Overview of the content negotiation ontology

*Bob* is interested in *Abies numidica*. To answer this request, *Bobajob* actively searches for potential web servers on *Search Agent*. Just like our use of search engines (e.g. DuckDuckGo and Google), however, such a search engine is for negotiable resources on the Web. Given a resource URI, such an agent is able to indicate whether that resource is negotiable or not, in what dimension, how to convey constraints, and how to negotiate for content.

In this scenario we also consider that web servers *YoucTagh Server* and *Lyncis Server* are managed by agents *YoucTagh SAgent* and *Lyncis SAgent* respectively. These agents may have goals such as detecting broken links and interacting with other agents. For example, they can inform *Search Agent* about the negotiable resources on the servers they manage by providing a description of each of them, mainly including the characteristics discussed in Section 3.3, so that other agents know how to engage in content negotiation with those servers.

If *Bobajob* does not receive an answer from *Search Agent* after several attempts, it relies on *Semantic Negotiator Agent*, an agent that can negotiate with the third-party mediator we introduced in Section 7.4 to find a representation of the same resource that is identified with different URIs.

## 8.3 Content Negotiation Characteristics Ontology

In this section we present *CNCO*, the Content Negotiation Characteristics Ontology for creating negotiable resource descriptions that reflect the content negotiation characteristics employed by the server proposing a resource identified by a URI. We have used OWL 2 [73] to create the ontology available at `http://w3id.org/cnco`.

Figure 8.1 depicts a simplified overview of *CNCO*. The ontology defines five classes, which are described in the following paragraphs:

- `cnco:NegotiableResource` is the class of negotiable resources, resources that have multiple representations (variants) associated with them and content negotiation can be engaged to select appropriate ones.
- `cnco:CnDimension` is the class of content negotiation dimensions that are the primary distinguishing factor between a set of alternative representations of a resource.
- `cnco:CnStyle` is the class of content negotiation styles that can be used when handling a request.
- `cnco:CnCcMeans` is the class of content negotiation constraint conveyance means that can be used to convey preferences.
- `cnco:CnProtocol` is the class of protocols that can be used to perform content negotiation.

In addition, we define four vocabularies: content negotiation dimensions, content negotiation styles, content negotiation protocols and content negotiation constraint conveyance means. The first three vocabularies provide instances of the classes `cnco:CnDimension`, `cnco:CnStyle` and `cnco:CnProtocol`, which can be used out of the box when describing negotiable resources. The latter provides concepts and properties for defining `cnco:CnCcMeans`.

`cndimension:profile` and `cndimension:mediatype` are examples of content negotiation dimensions (see Figure B.6). `cnstyle:proactive` and `cnstyle:reactive` are examples of content negotiation styles (see Figure B.8). `cnprotocol:http` and `cnprotocol:coap` are examples of content negotiation protocols (see Figure B.9).

Figure 8.2 shows the content negotiation constraint conveyance means vocabulary. It defines two subclasses of `cnccmeans:CnCcMeans`, namely `cnccmeans:HeaderBased` and `cnccmeans:UriBased`.

`cnccmeans:PathExtension`, `cnccmeans:ARK` and `cnccmeans:QSA` are all subclasses of `cnccmeans:UriBased`.

We also define two properties, `cnccmeans:usesHeader` and `cnccmeans:usesParam`. The former is used to define the required header when `cnccmeans:HeaderBased` is used, e.g. `accept` or `accept-profile`. The latter is used to define the required URI parameter when `cnccmeans:QSA` is used, e.g. `format` or `profile`.

Listing 8.1 shows an example of how to describe a negotiable resource using the *CNCO* ontology together with the four supporting vocabularies. In the example we state that the resource *cntf* identified by `http://w3id.org/cntf/ontology` is a `cnco:NegotiableResource` and therefore its representations are negotiable. The

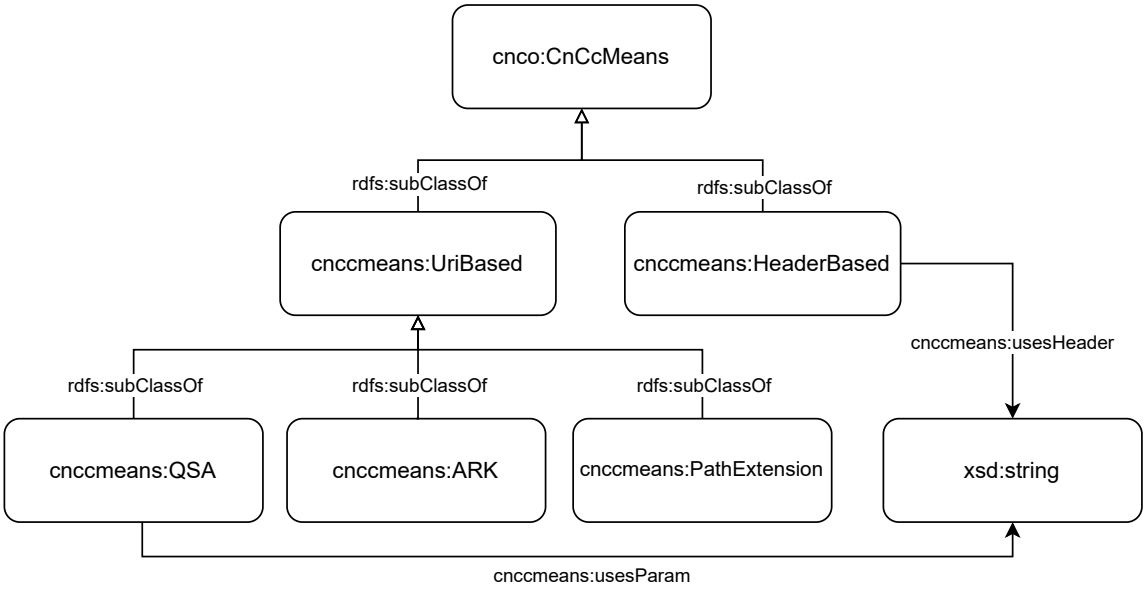



Figure 8.2: Overview of the content negotiation constraint conveyance means vocabulary

content negotiation dimension, style and protocol are profile, proactive and HTTP respectively. The header `accept-profile` should be used to convey the constraints.

## 8.4 Design – Achieving Content Negotiation in Web-based MAS

In this section we present the architecture used to create and simulate the scenario presented in Section 8.2. We design the agents' environment as a first-class abstraction [181], this layer provides the agents with functionalities to interact, communicate, coordinate and also act as a medium to access the external environment, for example servers on the Web. As shown in Figure 8.3, we have three layers:

First, the agent layer, where we can see the agents involved in the scenario: *Bobajob*, *YoucTagh SAgent*, *Lyncis SAgent*, *Search Agent* and *Semantic Negotiator Agent*. Secondly, the application environment layer, where we can find the artefacts that the agents use to interact with the third layer. The artefacts are organised in workspaces and for simplicity we have only one environment. Finally, the third layer is the external environment. In our scenario we have the two web servers if *YoucTagh* and *Lyncis* as well as the third party negotiator.

Listing 8.1: An example of a negotiable resource described using the content negotiation ontology

```
@base <https://w3id.org/cntf/ontology> .
...
<> a cnco:NegotiableResource;
    rdfs:label "cntf";
    rdfs:comment "The CNTF ontology";
    cnco:hasCnStyle cnstyle:proactive;
    cnco:hasCnDimension cndimension:profile;
    cnco:hasProtocol cnprotocol:http;
    cnco:hasCnCcMeans ex:header .

ex:header a cnccmeans:HeaderBased;
    cnccmeans:usesHeader "accept-profile" .
```

Each agent has its own beliefs, goals and actions. For example, *Bobajob* believes that it accepts the media types `text/html` and `text/plain` and has a goal to negotiate for the believed needed resource identified by some URI and can take the action to negotiate for a resource representation if a negotiable resource description is available. Also, artefacts propose operations to perform the actions and properties to observe, e.g. formulating an HTTP `GET` request with the appropriate `accept` header, sending it to the desired server, and reacting to the server's response accordingly.

The simulation of the scenario consists of the following phases. However, it should be noted that these phases are not all sequential, as the agents act asynchronously when appropriate:

- The agents and their relevant artefacts are created and initiated in the appropriate workspaces.
- *YoucTagh SAgent* informs *Search Agent* about the resources negotiable on the *YoucTagh Server* by providing the appropriate RDF representations.
- *Lyncis SAgent* asks *YoucTagh SAgent* for the name of the agent it will use to submit its negotiable resource descriptions.
- *Lyncis SAgent* informs *Search Agent* of the resources negotiable on *Lyncis Server* by providing the appropriate RDF representations.
- *Bobajob* asks *Search Agent* for a description of the negotiable resource *Abies numidica*.

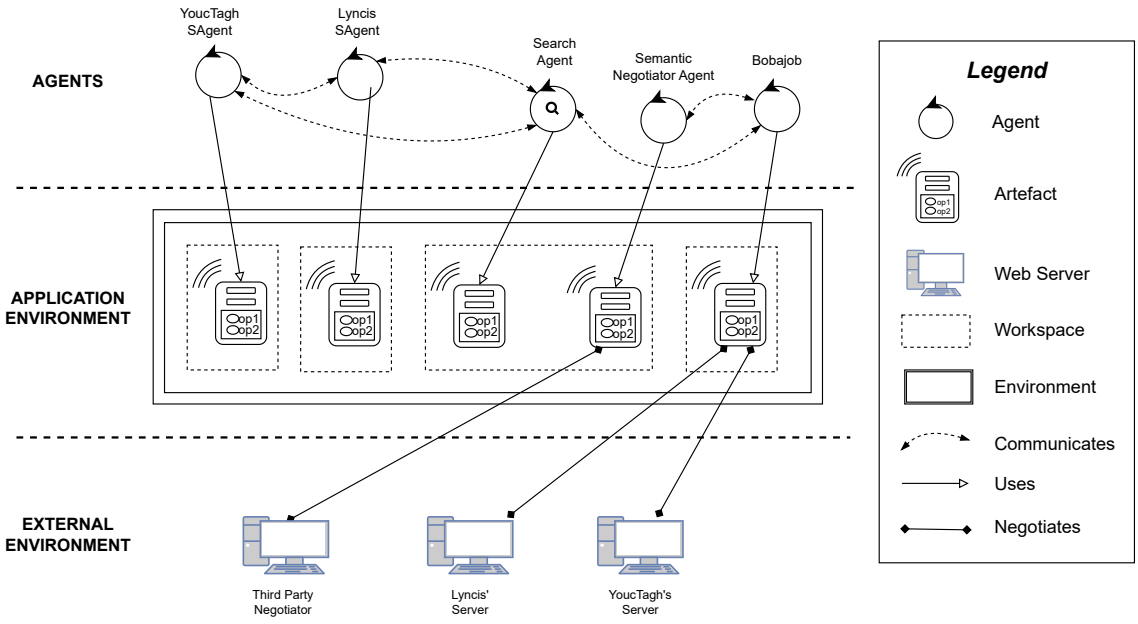



Figure 8.3: Overview of the Use Case 8.2 architecture

- *Search Agent* notifies *Bobajob* by providing the RDF representation describing the properties used by *YoucTagh Server* to negotiate the desired content.
- *Bobajob* uses the description of the negotiable resource to formulate the request and handle the negotiation interaction.
- In the case of several failed attempts to request a description of the negotiable resource *Abies numidica*, *Bobajob* asks *Semantic Negotiator Agent* to negotiate a representation of that resource on its behalf.
- *Semantic Negotiator Agent* asks *Bobajob* for a preferred profile if needed.
- *Semantic Negotiator Agent* formulates, negotiates and gives the final answer to *Bobajob*.

## 8.5 Practical Implementation – Achieving Content Negotiation in Web-based MAS

In this section we present how we implemented the use case presented in Section 8.2 using the JaCaMo platform [27].[1] We have developed a Java library for *CNCO*. The library allows among other things to manipulate RDF graphs of negotiable resource descriptions by reading and writing such representations. The Java library *rdf4j*[2] has been used to manipulate the graphs. Listing 8.2 shows a snippet of the creation and writing of the RDF graph presented in Listing 8.1. First, we create an instance of the `NegotiableResource` class. Then we set its identifier, label and description as a comment. Then we set its content negotiation style, dimension and protocol. Finally, we set how to convey constraints when negotiating this resource, as well as the appropriate header or URI parameter if needed. Note that if no header or parameter is set, we use default ones, e.g. `accept` for media type and `accept-profile` for profile.

Listing 8.2: The creation of a negotiable resource description using the *CNCO* library

```
1  NegotiableResource cntf = (NegotiableResource)
2    new NegotiableResource()
3      .setLabel("cntf")
4      .setComment("The CNTF ontology")
5      .setIRI("https://w3id.org/cntf/ontology");
6  cntf.setCnStyle((CnStyle) new CnStyle()
7    .setIRI(TERM.CN_STYLE_PROACTIVE.toString()));
8  cntf.setCnDimension((CnDimension) new CnDimension()
9    .setIRI(TERM.CN_DIMENSION_PROFILE.toString()));
10 cntf.setCnProtocol((CnProtocol) new CnProtocol()
11   .setIRI(TERM.CN_PROTOCOL_HTTP.toString()));
12 cntf.setCnCcMeans((CnCcMeans) new CnCcMeansHeaderBased()
13   .setUsedHeader("accept-profile")
14 ...
```

Listing 8.3 shows a snippet of the JaCaMo application file for creating the MAS agents and environment. This file defines the initial state of the MAS following the architecture shown in Figure 8.3. For example, we create the agent *Bobajob* as well as the workspace that the agent will join. The workspace contains a negotiator artefact that the agent will focus on. This artefact provides some operations that the agent

[1] JaCaMo stands for Jason - CArtAgO - Moise, where Jason is used for programming autonomous agents, CArtAgO for the shared environment, and Moise for agent organisations. In our work we did not use the organisation part.

[2] rdf4j project home page: https://rdf4j.org/

can use when needed.

Listing 8.3: A snippet of the JaCaMo application file for creating the MAS agents and environment

```
mas jacamo_cn {
  agent bobajob: bob.asl {
    join: bob_workspace
    focus: bob_workspace.bob_client
  }
...
  workspace bob_workspace{
    artifact bob_client:
      fr.minesstetienne.ci.cn.NegotiatorArtifact("Bob")
  }
}
```

Listing 8.4 shows a snippet of some of the agent *Bobajob* initial beliefs, such as the acceptable media types and a URI to the profile to be used when negotiating in the profile dimension. Additionally, the URI to a needed resource.

Listing 8.4: A snippet of some of the agent *Bobajob* initial beliefs

```
acceptedValue("https://w3id.org/cnco/dimension#mediatype"
  ,"text/html").
acceptedValue("https://w3id.org/cnco/dimension#mediatype"
  ,"text/plain").
acceptedValue("https://w3id.org/cnco/dimension#profile"
  ,"http://path-to-prof.com/1").

neededResource("http://youctagh.com/species/abies_numidica").
```

Listing 8.5 shows a snippet of *Bobajob*'s *negotiate* plan, which is used to achieve its initial goal by repeatedly trying to negotiate the needed resource if the maximum number of attempts has not yet been reached.

Listing 8.5: A snippet of *Bobajob*'s *negotiate* plan

```
+!negotiate(C,R) : C > 0 <-
  .send(kg_mediator,askAll,canNegotiate(Who,R,Rgraph));
  .print("ask mediator to know with whom negotiate");
  .wait(5000);
  decAttemptCount;
  getAttemptCount(NewCount);
  !negotiate(NewCount,R).
```

Figure 8.4 shows a snapshot of the agent *Search Agent* beliefs after receiving the negotiable resources descriptions from the other agents.

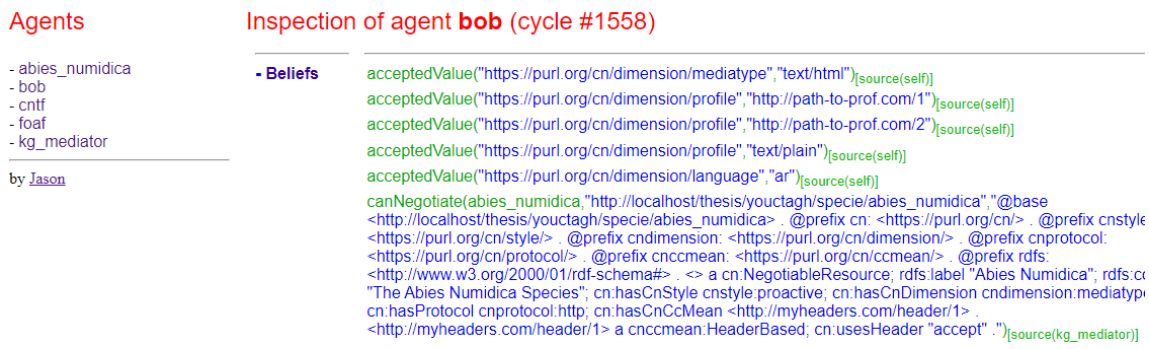



Figure 8.4: A snapshot of the agents *Search Agent* beliefs after receiving the negotiable resources descriptions

Listing 8.6 shows the initialisation of the artifact used by *YoucTagh SAgent*. *CNCO* library is used to indicate a local file path of the negotiable resource description, the implementation could be adapted to accommodate indicating remote URIs.

Listing 8.6: A snippet of the initialisation of the artifact used by *YoucTagh SAgent*

```
1 ...
2 try {
3   this.absoluteLocation = getAbsoluteLocation(location);
4   this.negotiableResource = NegotiableResourceReader
5     .readFromFile(absoluteLocation);
6 }
7 ...
```

Listing 8.7 shows the operation *negotiateARepresentation* offered by the artefact *NegotiatorArtifact* used by the agent *Bobajob*. First, a graph of a negotiable resource is read, such as the one shown in Listing 8.1. Then a request is formed according to this description. For example, to negotiate a representation of the *cntf* resource, an HTTP `GET` request is created with the `accept-profile` header and the URI `http://path-to-prof.com/1` as its value (as specified in the agent's beliefs, see Listing 8.4 lines 5-6).

Listing 8.7: A snippet of an operation offered by an artefact

```
1 @OPERATION
2 public void negotiateARepresentation(String resourceKG) {
3   NegotiableResource negotiableResource = NegotiableResourceReader
4     .readFromString(resourceKG);
5 ...
```

```
6    CnCcMeans cnCcMeans = negotiableResource.getCnCcMeans();
7    HttpURLConnection con = null;
8    if (cnCcMeans instanceof CnCcMeansHeaderBased) {
9      URL obj = new URL(url);
10     con = (HttpURLConnection) obj.openConnection();
11     con.setRequestProperty(((CnCcMeansHeaderBased) cnCcMeans).
           getUsedHeader().get(),
12       acceptedValues.stream().reduce("", (a, b) -> a + b + ","));
13 ...
```

A custom *JFrame* is provided by *Bobajob* to display for example the negotiation results. Figure 8.5 shows a screenshot of the JaCaMo console as the simulation unfolds.

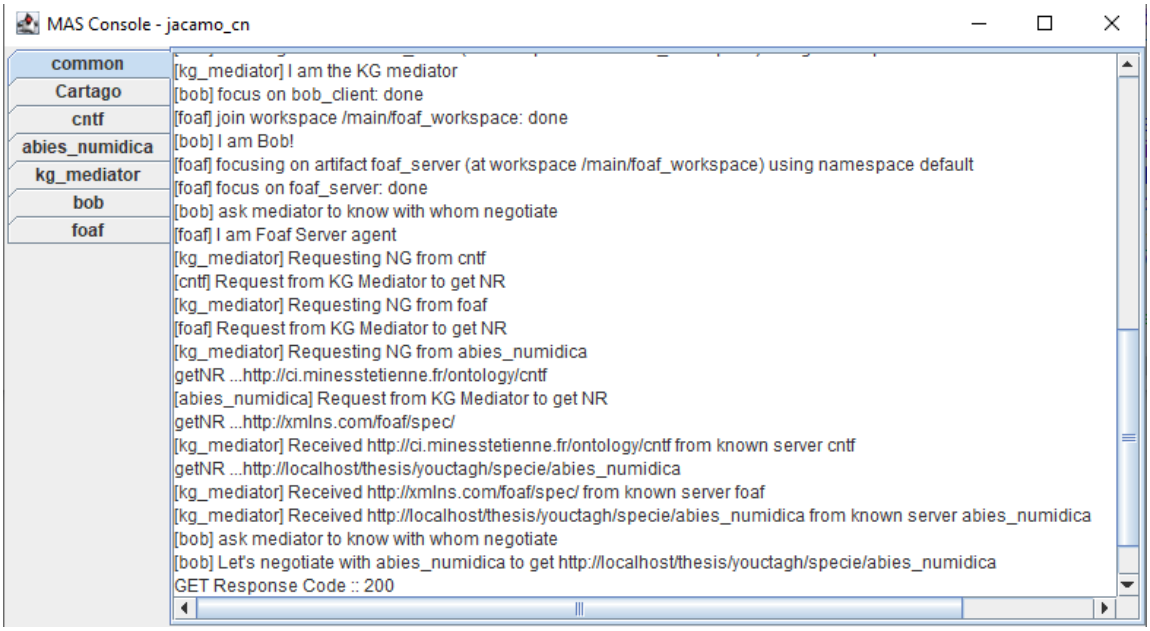



Figure 8.5: A screenshot of the JaCaMo console showing the simulation results

For example, in this scenario we see that the first attempt was unsuccessful. However, the second negotiation resulted in a `200 OK` and a successful negotiation of the resource using the negotiable resource description.

## 8.6 Summary

In this chapter we have addressed Research Question 5, which is *How can we enable autonomous agents to discover and exploit negotiable resources?*.

In Section 8.1, we provided an overview of content negotiation in MAS. Starting from MAS, we moved on to the new vision of autonomous systems on the Web, called hypermedia MAS. We have seen that there is a need to enable autonomous software agents to engage in semantic content negotiation to request tailored and appropriate content. In Section 8.2, we explored a use case in a MAS setting where negotiable resources are needed, but agents have no prior knowledge of the characteristics used by the web servers that propose the resource representations. Also, since different agents accept different semantics, semantic content negotiation is required and employed. In Section 8.3, we introduced *CNCO*, the ontology that enables the capture of knowledge about the content negotiation characteristics of web servers. In section 8.4 we presented the steps needed to achieve and design such a MAS, and then in Section 8.5 we discussed the practical implementation to simulate the use case and show how its requirements were met.

This chapter concludes the third part of this thesis. In the next part, we will summarise our work, highlight the different contributions and provide perspectives for future research.

# Part IV

# Conclusions and Perspectives

Chapter 9

# Conclusions and Perspectives

## Contents



The goal of this thesis is to contribute to the development of a semantic content negotiation framework in which web agents engage in a more *flexible* and *relaxed* content negotiation, taking into account that this process is not monolithic, but consists of several stages: (1) discovery; (2) formulation of the request; (3) selection and/or adaptation; (4) indication of the response; (5) interpretation of the response.

We proposed to achieve this goal by using semantic technologies and tooling to improve content negotiation in several aspects: (1) discovering potential representations through equivalence links, in addition to discovering negotiable resources through their descriptions, which uses the *CNCO* ontology we proposed; (2) indicating finer constraints through severity and knowing how to engage in content negotiation through the negotiable resource descriptions; (3) relaxing validation; (4) reporting finer validation; (5) interpreting the validation report and reacting to the response according to the negotiable resource descriptions.

In this chapter, we provide a summary of this dissertation, our contributions, and discuss perspectives for future research.

## 9.1 Summary

In Chapter 1, we presented an overview of our thesis problem, the limitations we identified, and the research questions we addressed in our work. This dissertation is structured in four parts.

In Part I, we analysed the state of the art to highlight the beginning and evolution of content negotiation on the Web, and searched for related work and technologies

that could be used to bring about the envisioned flexible and relaxed semantic content negotiation framework.

In Chapter 2, we provided the background knowledge necessary for the rest of the thesis. We began by introducing the content negotiation problem in the light of the emergence of the Web. We then reviewed the Semantic Web, the extension of the Web that gives structure and meaning to web content, and a selection of its components used throughout the thesis.

In Chapter 3, we discussed in more details the limitations that motivate our work. First, we reviewed the different stages of content negotiation and examined how previous work has attempted to formalise it. To the best of our knowledge, there is no existing formalisation that reflects the current practice of doing content negotiation in the common dimensions, e.g. media type, and is extensible in the Semantic Web context, e.g. profile dimension. We then discussed the different content negotiation characteristics (i.e. dimensions, styles and constraints conveyance means) with a closer look at the profile dimension, focusing on two main questions: how can a client specify a profile? and how can preferences be conveyed in this content negotiation dimension? However, while content negotiation styles are considered general patterns, there are no finer patterns that shape the negotiation process, especially in the profile dimension. Some of the previous W3C and IETF efforts had focused on content negotiation in this particular dimension, but no proposal had been made to relax the process or make it more flexible.

In Part II, we presented our contributions towards designing a flexible and relaxed content negotiation.

In Chapter 4, we presented CON-NEG, our proposed general model for content negotiation. We first presented a general case that reflects current practices, and discussed how it fits into the way of doing content negotiation in HTTP. We then showed how we extended the general model in the context of Semantic Web to allow content negotiation in the profile dimension.

In Chapter 5, we described how we improved the content negotiation process by utilising semantic validation, guided by the content negotiation stages: first by specifying finer constraints, then by relaxing the validation and reporting finer validation, and finally by interpreting the validation report.

In Chapter 6, we discussed possible finer patterns for handling content negotiation in the profile dimension. In particular, we showed how some of them can take advantage of current web standards and infrastructure. These finer patterns were

illustrated using an incremental use case and the current efforts of the DXWG and IETF.

In Part III, we presented the validation of our work in practical settings.

In Chapter 7, we explored the *representation distribution problem* in more detail, in which multiple servers may have different representations of the same resource, and thus a client negotiating with one of them is only considering a subset of the total available possibilities. Then, we extended CON-NEG to propose a formalisation for this problem. After that, we presented an approach to perform content negotiation when representations are distributed by leveraging equivalence links involving on-the-fly conformance checking. To this end, we provided two algorithms, the first in a basic context (i.e., considering only media type constraints) and the second in a semantic context (i.e., considering profiles in addition). The implementation of the algorithms, as well as the separate experiments that were conducted, measured the benefits and assessed the time requirements of the proposed methods.

In Chapter 8, we introduced our contribution towards facilitating the interaction of heterogeneous systems. First, we had a look at recent related results from multi-agent research, where we found that there are many existing paradigms and technologies relevant to validating our envisaged flexible and relaxed content negotiation in such systems. However, in the context where no prior knowledge of the content negotiation characteristics used by web servers is hard-coded, a formal, general and reusable means of expressing such knowledge is needed. Then, we described a Web-based Multi-Agent Systems use case where semantic content negotiation is needed. Then we presented *CNCO*, the content negotiation ontology. After that, we illustrated the steps to design and implement such an environment, where *CNCO* is used to provide negotiable resource descriptions that are used by agents to achieve their goals requiring semantic content negotiation.

## 9.2 Review of Main Contributions

We summarise and organise our contributions according to the five research questions we address in this dissertation:

**Research Question 1.** *How can we formalise the problem of content negotiation in a general and extensible way to accommodate negotiation by profile?*

We proposed CON-NEG, a general abstract model for content negotiation (see Section 4.2.1), where we chose the right level of abstraction and the elements to in-

clude in the abstract model in a way that makes it inclusive and extensible for future use. We demonstrated the extensibility of our model to a semantic context to allow the negotiation of representations conforming to some profiles (see Section 4.2.2). We proposed S-CON-NEG, a model for flexible and relaxed content negotiation (see Section 5.3). We discussed the applicability of our models CON-NEG and S-CON-NEG in Section 4.3 and Section 5.4 respectively. Parts of this proposal were presented at the 20th Meeting of Young Researchers in Artificial Intelligence (RJCIA 2022) [168].

**Research Question 2.** *How can a client and a server engage in a relaxed content negotiation process?*

We proposed our approach to enhance the content negotiation process by making it more flexible and relaxed, while being guided by the content negotiation stages and following the general architectural principles of the Web (see Section 5.5). To this end, we provided solutions for specifying finer constraints through the use of relaxed constraints. Then, at the server side, we proposed to relax the validation by taking into account these finer constraints and using scoring methods to select appropriate representations and report the validation results back to the client, which in turn can interpret and make use of such improved responses (see Section 5.2). We demonstrated the applicability of our approach in Section 7.4.2, where a client is able to request fine-grained constraints and a server takes them into account when selecting the best available option, especially in the case of non-conformance of optional constraints.

Parts of this proposal were presented at the 23rd International Conference on Knowledge Engineering and Knowledge Management (EKAW 2022) [166], and at the 20th Meeting of Young Researchers in Artificial Intelligence (RJCIA 2022) [168].

**Research Question 3.** *What are the finer patterns that can shape the negotiation process in the profile dimension?*

We identified, collected and classified finer patterns used when negotiating content in the profile dimension (see Section 6.1). These finer patterns answer more precise questions than content negotiation styles, especially when flexible and relaxed content negotiation is being adopted. We illustrated how such finer patterns can be used to classify existing implementations (see Section 6.2). We demonstrated, through the creation of a mediator (a third party negotiator), a combination of content negotiation finer patterns that differ from the existing implementations already available (see Section 7.4.1). Parts of this proposal were presented at the 23rd International Conference on Knowledge Engineering and Knowledge Management (EKAW 2022) [166].

**Research Question 4.** *How can we negotiate an appropriate variant of a resource when its representations are distributed?*

We proposed a formalisation of the distributed content negotiation problem based on our model CON-NEG. We proposed the use of equivalence links to discover potentially acceptable representations (see Section 7.3). We proposed two algorithms that used our approach for two content negotiation dimensions (media type and profile in Section 7.3.1 and Section 7.3.2 respectively). We demonstrated the applicability of our approach in Section 7.4.2, where we relied on a third-party service to discover potentially acceptable representations. Then, in Section 7.5, we verified the hypothesis that using such a technique increases the rate at which an acceptable representation is found. Parts of this proposal were presented at the 1st International Workshop on Data Management for Knowledge Graphs (DMKG 2023) at ESWC [169], as well as at the 2nd workshop of the doctoral college for Semantics, Reasoning and Coordination (SeReCo 2023) [165].

**Research Question 5.** *How can we enable autonomous agents to discover and exploit negotiable resources?*

We proposed *CNCO*, an ontology for describing negotiable resources available at web servers (see Section 8.3). We introduced supporting vocabularies for *CNCO* for already known content negotiation dimensions, styles and constraint conveyance means (see Section 8.3 and Appendix B). Producers of negotiable web resources can use these vocabularies out of the box to create negotiable resource descriptions. We demonstrated the applicability of our approach in Section 8.4, where we reused existing multi-agent models and technologies. Then, we simulated a scenario where software agents use negotiable resource descriptions to know how to engage in content negotiation with previously unknown web servers (see Section 8.5). Parts of this proposal were presented at the 23rd International Conference on Knowledge Engineering and Knowledge Management (EKAW 2022) [166].

## 9.3 Future Work

In this section, we outline some of the prospects that our research opens up for future investigations:

First, in our work we have focused mainly on HTTP `GET` requests. But note that, as discussed in Chapter 3, content negotiation can also be performed in `POST`, for example, where the client and server negotiate the accepted content to be posted. The

request/response model has also been used to illustrate our contributions. Thus, one perspective is to experiment with the publish/subscribe model [68] and explore the implications of using such flexible and relaxed content negotiation in these contexts.

Second, in our survey of content negotiation finer patterns we have shown that different combinations have been proposed and used over the years, one of which is ours. One perspective of this work is to perform an experimental comparison of different finer patterns in terms of performance, complexity and availability of representations.

Third, in our formalisation we have focused mainly on the content negotiation proactive style. One perspective is to extend it to other styles. For example, in the reactive style, where it is rather the client that makes the selection decision, the CN measure should not be part of the server data. The same apply if we wish to formalise HTTP redirections.

Fourth, in our work we did not investigate in depth the case of engaging in flexible and relaxed negotiation of adapted content. One direction, for example, is to rely on the repair of SHACL constraint violations. Research that uses answer-set programming [3] might be a good starting point.

Fifth, CNTF aims to highlight existing solutions for content negotiation use cases, where available, or suggest plausible ways to meet common requirements (see Appendix A). We plan to have an implementation of each approach explored and discussed in this dissertation to facilitate their adoption by having an experimental space. Such a space would allow users to test different content negotiation styles, dimensions, and solutions to some of the use cases without having to implement them themselves.

Finally, another related perspective is to propose a content negotiation validator service similar to Vapour [24] and Memento validator [116] but with additional content negotiation dimensions. The service in addition would create a negotiable resource description that the person validating can publish if they so choose. And if the owner gives permission, that description can be added to a KG which, ultimately, in the long run, can be used by a global negotiable resource search engine similar to the one presented in Chapter 8.

# List of Publications

Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Negociation de contenu sur le web: un état de l'art. In: Journées Francophones d'Ingénierie des Connaissances (IC) PFIA (Jun 2022).

Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Content Negotiation on the Web: State of the Art. Tech. Rep. abs/2204.10097, CoRR (Apr 21 2022).

Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Négociation de contenus sémantique pour l'échange de connaissances entre systèmes hétérogènes. In: 20èmes Rencontres des Jeunes Chercheurs en Intelligence Artificielle (RJCIA) PFIA (Jun 2022).

Taghzouti, Y., Vachtsevanou, D., Mayer, S., Ciortea, A.: A Step Toward Semantic Content Negotiation. In: Companion Proceedings of the 23rd International Conference on Knowledge Engineering and Knowledge Management, BozenBolzano, Italy, September 26-29, 2022. CEUR Workshop Proceedings, vol. 3256. CEUR-WS.org (Sep 2022).

Taghzouti, Y., Zimmermann, A., Lefrançois, M.: A Portal for the State of the Art on Content Negotiation. In: Proceedings of the ISWC 2022 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 21st International Semantic Web Conference (ISWC 2022), Virtual Conference, Hangzhou, China, October 23-27, 2022. CEUR Workshop Proceedings, vol. 3254. CEUR-WS.org (Oct 2022),

Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Content Negotiation in a Decentralised Semantic Context Utilising Equivalence Links. In: DMKG Workshop at the ESWC 2023. CEUR-WS.org (2023)

Taghzouti, Y. Enable Decentralised Semantic Content Negotiation through Equivalence Links. In: SeReCo Summer Workshop 2023, Waischenfeld, Germany. (Jul 2023)

# Part V

# Appendices

Appendix A

# Content Negotiation Theoretical Framework

## Contents



Over the years, various dimensions of CN, styles of CN, as well as ways to convey constraints have emerged to satisfy new requirements and solve new use cases. This appendix presents a new resource called Content Negotiation Theoretical Framework (CNTF): a website that collects knowledge about CN use cases, styles, dimensions, etc., and organise it according to an ontology. CNTF aims to highlight existing solutions if available, or suggest plausible ways to satisfy common requirements. It is intended to be used to disseminate our future proposals for advancing CN, making it a sustainable and up-to-date digital survey of CN. The idea of a digital state of the art is not new; others have already proposed similar resources, either in the form of Web pages, or in the form of a website dedicated to a topic like the one for the complexity of reasoning in logic of description http://www.cs.man.ac.uk/~ezolin/dl/.

This proposal was presented at the 21st International Semantic Web Conference (ISWC 2022). The state of the art was presented at the 33rd Francophone Days of Knowledge Engineering (IC 2022) [164] and was translated and published in ArXiv [170]

## A.1 Use cases and Requirements for CNTF

In this section, we discuss the potential impact of CNTF as a resource. To this end, we present different use cases that motivate its creation. We highlight the challenges

that these use cases reveal, and obtain a set of requirements for CNTF to address these challenges.

**Use case 1** Jane is a researcher and John is a developer, they are both interested in CN. While Jane is preparing a PhD in relation to that subject, John has a task to develop a CN mechanism for its company to serve different archived versions of their content served via an API. Both Jane and John *have to search different websites* for the old methods of CN or the most modern ones (C1). It would also take them *ages to come up with a coherent understanding of the relationship between the different approaches* (C2), especially because *one term may have different meanings in different communities of researchers*. One of the goals of CNTF is to provide a digital catalogue of everything one needs to know about content negotiation.

**Use case 2** Paul has an idea for a new approach to do CN, and although he already has a use case to motivate it, he wants to compare it with existing approaches, and to know what style, dimension, and negotiation requirement his use case fits into. For now, the lack of a resource categorising the different concepts of CN makes it difficult to compare with other approaches. The task of Paul would be eased by *a categorisation and grouping medium* (C4) and *assistance and guidance to select a classification* (C5). CNTF aims to address these.

**Use case 3** Alice is a semantic web researcher involved in a project for which there is a plan to use CN and she is interested in learning more about this topic. While UC#1 contains resources and data, UC#2 allows for grouping and categorisation. As a semantic web scholar she would be interested in *a model of how everything is related, with each of the content negotiation concepts represented with all of its relationships* (C6). Also, *a visual explanation for perceiving these relationships if possible* (C7), and *a procedure for graphically navigating from one concept to another* (C8). This is what CNTF intends to provide.

From these use cases, we identify requirements that the CNTF should meet:

$\mathbf{R}_1$ **Navigable design** The CNTF should have a navigable design that includes interlinking through hyperlinks to help users categorise their use cases (C5), and a graph following the resource model (C7), with clickable nodes to navigate from one concept to another (C8).

$\mathbf{R}_2$ **Extensible** One of the main contrasts between a traditional survey paper and the CNTF resource is that it should supports extensibility by allowing the

addition of new content negotiation concepts, e.g., a new dimension (C1), and to be up to date with different terminology and definitions (C3).

**$R_3$ Categorisable** The CNTF should provide the means to categorise the different content negotiation use cases and techniques (C4) to allow for comparative evaluation. It should also have a well thought out grouping (C2) and modelling (C6) of the different content negotiation concepts to facilitate understanding.

**$R_4$ Maintainable** The CNTF should promote maintainability by adjusting the model used, e.g., community-recommended vocabularies (C6), and by taking into account feedback provided by resource users to clarify and rectify content (C1).

Table 1 summarises the relationship between the challenges (Ci) and the requirements (Rj).

| Use case | Use case 1 | | | Use case 2 | | Use case 3 | | |
|---|---|---|---|---|---|---|---|---|
| Challenge / Requirement | C1 | C2 | C3 | C4 | C5 | C6 | C7 | C8 |
| $R_1$: Navigable design | | | | | ✓ | | ✓ | ✓ |
| $R_2$: Extensible | ✓ | | ✓ | | | | | |
| $R_3$: Categorisable | | ✓ | | ✓ | | ✓ | | |
| $R_4$: Maintainable | ✓ | | | | | ✓ | | |

Table A.1: Relationships between the challenges (Ci) and the requirements (Rj)

## A.2 A Portal for the State of the Art on Content Negotiation

CNTF is a website designed to help the web community: either newcomers to the field of content negotiation who want to explore it, or indigenous who want to keep up with the latest techniques.

In this section, we present the architecture behind CNTF, followed by the accompanying features and functionalities, and show that it meets the requirements presented in Section A.1.

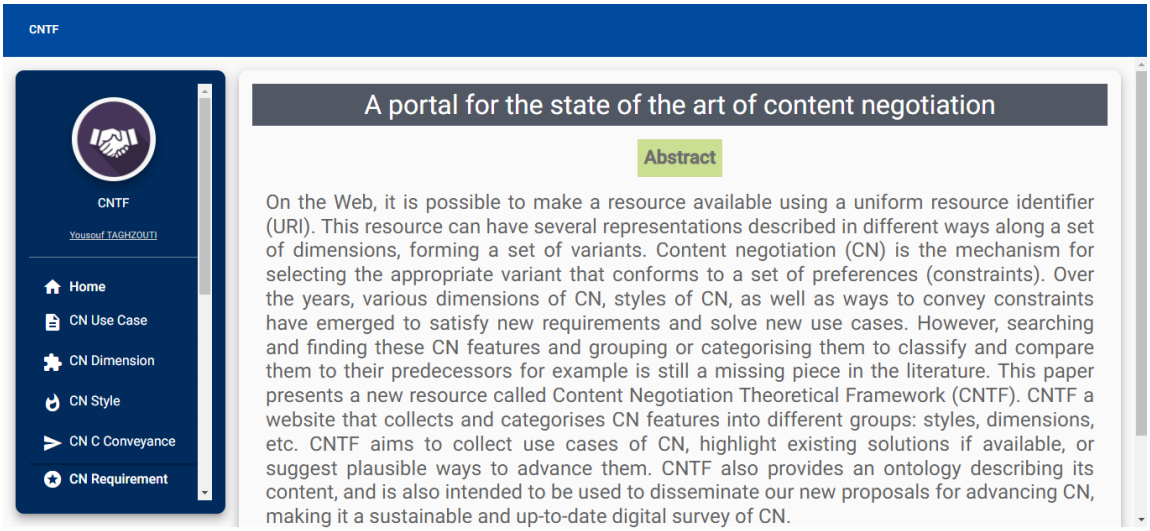



Figure A.1: CNTF home page.

The system has been designed as a website, so no installation process is required. In addition, it allows users to always have the latest version without having to perform an update at each release. CNTF is built with the Angular Framework[1] with cross-browser compatibility. Figure A.1 shows the home page of CNTF.

## CNTF as a State of the Art Literature Review of CN

As mentioned in the introduction content negotiation could be achieved using different styles, CNTF collects these styles and gives:

- A description of the style.
- An example of content negotiation using that style.
- The use cases from the bank of use cases in CNTF using that style.
- Pointing the advantages and disadvantages of this style.
- Relevant references where the style was introduced or mentioned.

Over the year the preferences of the client/server in content negotiation have been referred to with different designations “constraints”, “features” etc, in CNTF

[1] https://angular.io/

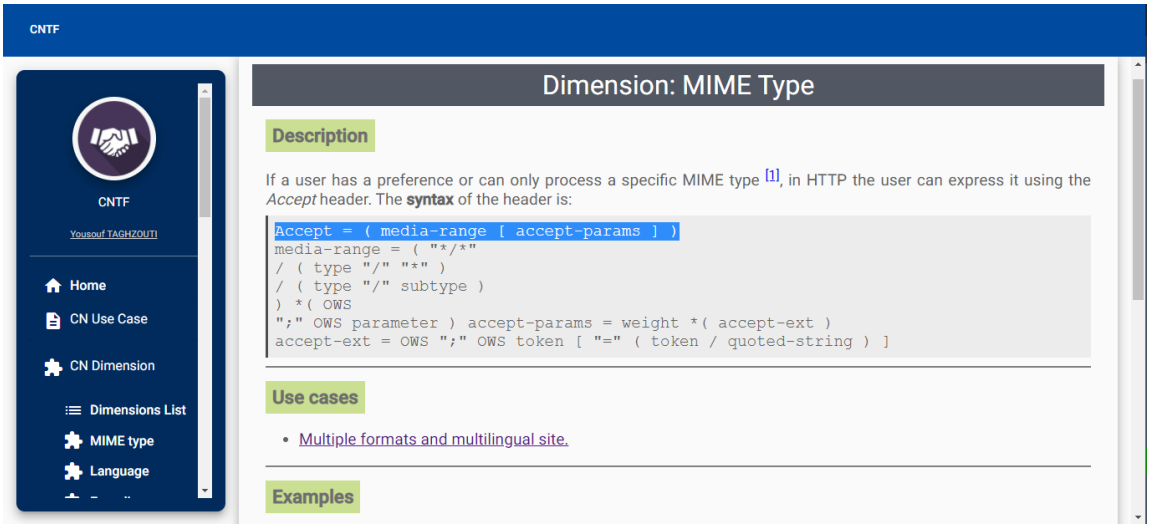



Figure A.2: CNTF MIME dimension example.

we group them in a category named content negotiation dimension and for each of these dimensions we deliver:

- A description of the dimension.
- Some references to the pertinent use cases.
- Some examples of the content negotiation process.
- Some relevant references on the state of the art.

Each domain comes with a set of terms, for a newcomer to find and grasp the definitions of all these terms is a tedious task, that's why CNTF has a terminology section that lists and keeps track of the meaning of the terms related to content negotiation, and since different communities might differ on the meaning of a term, we present them with their respective communities references. Also, since some terms are prone to confusion, CNTF has a dedicated section to provide more clarity, e.g. "Media type vs MIME type vs format".

Since another type of users of this resource is developers, the CNTF has an "Additional Info" section that lists implementations of content negotiation using some of the available technologies. Figure A.2 represents a CN dimension example.

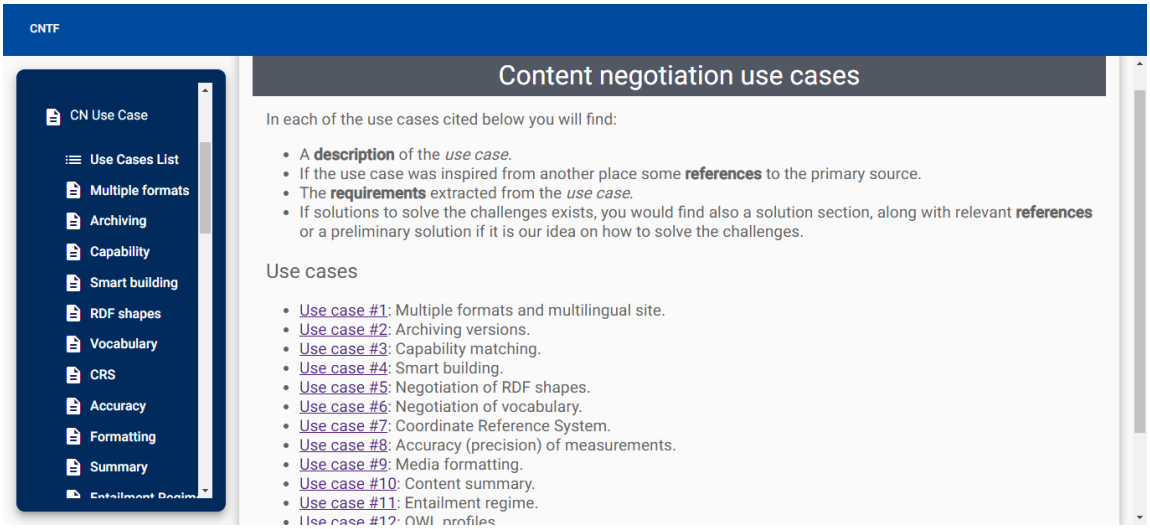



Figure A.3: CNTF use case list page.

## CNTF as a Bank of CN Use Cases

A difficulty faced by someone making research in a new subject area is to contextualise the usefulness of the material learned. CNTF attempts to solve this problem for the CN domain by collecting use cases from other documents and websites. Being an open-source project, the community may contribute with other use cases. In each of the use cases one finds:

- A description of the use case.
- Potentially some references to the primary source the use case was inspired from.
- The requirements extracted from the use case.
- Potentially a solution section with relevant references or preliminary solutions.

Each of these use cases must satisfy a set of requirements that could be found in the "CN requirement" section. Figure A.3 shows the use case list page.

## CNTF as a Categorisation and Classification Medium

CNTF is organised in categories, which favours categorisation and classification. A user may check all the concepts of content negotiation: styles, dimensions, etc.,

| Ref | Date | Style | Dimension | Transmission | Protocol |
|---|---|---|---|---|---|
| [178] | 2018 | Reactive | Media type | ARK | HTTP |

Table A.2: An example of the classification of a CN contribution

and select those that best fit his/her use case. A table with the same structure as Table A.2 is found in the "Classification" section in CNTF, where each row of the table is accompanied by a summary to aid understanding. For example the row in Table A.2 has the summary:

> "Istex is a French scientific archive [178]. The clients have the possibility to have the representation in several formats and for that, as mentioned on the site, an ARK is used. If we request the resource without specifying the media type, we receive a JSON file describing the existing variants and therefore we can consider it a reactive negotiation. The dimension is the media type and the transmission in ARK using the HTTP protocol."

## CNTF as a CN Knowledge Graph

Most of the content of CNTF is available in a knowledge graph (KG),[2] describing among other: use cases, references, dimensions, requirements. the CNTF knowledge graph is modelled according to the "CNTF ontology" https://w3id.org/cntf/ontology, with an excerpt in Listings A.1.

Listing A.1: Excerpt of the CNTF ontology https://w3id.org/cntf/ontology

```
@prefix : <https://w3id.org/cntf/ontology#> .

: a owl:Ontology;
  dct:title "CNTF ontology"@en;
  dct:description "This is the ontology for the CNTF resource"@en
      .

:UseCase rdf:type owl:Class ;
         rdfs:comment "The class of use cases for Content
            Negotiation."@en ;
         rdfs:label "Use Case"@en .

:hasDimension rdf:type owl:ObjectProperty ;
```

[2] https://w3id.org/cntf/kg

```
12                rdfs:subPropertyOf owl:topObjectProperty ;
13                rdfs:domain :UseCase ;
14                rdfs:range :Dimension ;
15                rdfs:comment "Links a Content Negotiation use case
                     to a dimension of Content Negotiation"@en .
16
17 :hasRequirement rdf:type owl:ObjectProperty ;
18                  rdfs:subPropertyOf owl:topObjectProperty ;
19                  rdfs:domain :UseCase ;
20                  rdfs:range :Requirement ;
21                  rdfs:comment "Links a Content Negotiation use case
                        to one of its requirements."@en .
22 ...
```

## CNTF as a Medium for Visualisation and Navigation of CN Concepts

CNTF users can navigate with the traditional hyperlinks to move from one page to another in order to explore different concepts. Alternatively, the user can get a general view of the relationships between concepts through the CNTF ontology "visualisation" section.[3] WebVOWL [113] is used to render the visualisation as shown in Figure A.4.

## CNTF as a Template for State of the Art Resources

CNTF is a website that represents a digital form of the survey on the CN domain but some components that represent generic parts could be reused to fit other domains such as: Use Cases, Requirements, Classifications (change the table columns with the used metrics), Terminologies, Abbreviations, References, etc., making it a candidate to be a model for digital surveys. Moreover, the same parts could be reused from the ontology.

[3] https://w3id.org/cntf/visualisation

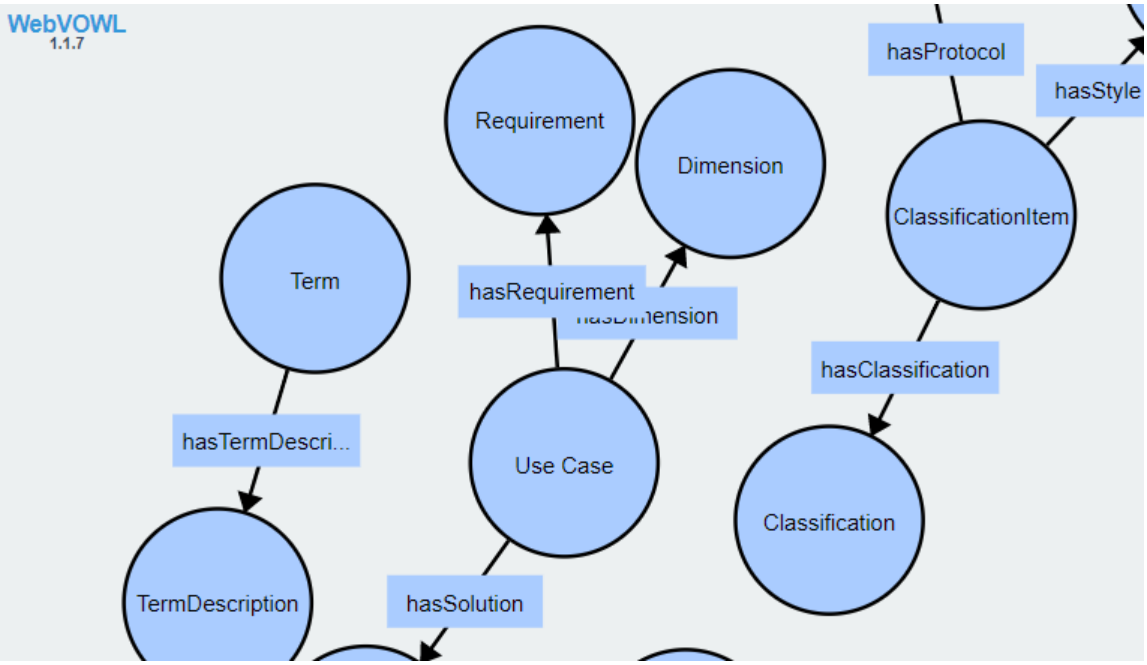



Figure A.4: A partial view of the CNTF ontology page.

## A.3 Relevance of CNTF for the Semantic Web Community

### Impact

CNTF reviews the available literature and compiles it into a single resource so that users interested in CN can easily find relevant materials. It addresses the problem of CN and attempts to identify new ways to better solve the available use cases. The resource covers the basics of CN: use cases, dimensions, styles with the advantages and disadvantages of using one over the other, and existing ways of conveying constraints. One can compare it to other resources dealing with CN, such as: MDN Web Docs,[4] Wikipedia,[5] or CN implementations in specific technologies and contexts [193, 195], each of these resources mentions only some of the features listed above, but the CNTF brings them together and links them to give a coherent view of the problem that will be the basis for addressing the semantic CN problem. Another example of its use is that if a developer finds that the use case under consideration matches one

[4] https://developer.mozilla.org/en-US/docs/Web/HTTP/Content_negotiation
[5] https://en.wikipedia.org/wiki/Content_negotiation

of those identified in the resource, the proposed solutions could be used to provide a better service and, in turn, increase end-user satisfaction. Finally, recommending solutions that leverage semantic web technologies for the collected use cases would generate interest in using semantic web technologies, e.g., leveraging SHACL as a way to express the threshold of CN constraints.[6]

## Reusability

Explanations are provided in the resource itself to aid user browsing. A concise documentation is also provided in GitHub for standard handling of the source code. The user can reuse some components in other state of the art digital resources as explained in the Section A.2. CNTF has a potential for extensibility to meet future requirements. Thanks to the component architecture, one can add a navigation bar menu item and then create a component with the desired functionality.

## Design & Technical quality

CNTF proposes an ontology describing its content, and reuses higher-level ontologies such as: `Dublin Core`, `FOAF`, `Biblio` etc. CNTF as illustrated in Section A.2 is able to meet the use cases and requirements described in Section A.1. Finally, the content of CNTF is by default available in a human-readable format (HTML) and also in a machine-readable form (RDF).

## Availability

CNTF is available at a persistent URI using w3id.[7] Citations are given in the GitHub[8] repository in BibTex or RDF along with the creators contact, CNTF is under the GNU General Public License v3, and the source code is available in the GitHub open code repository and in Zenodo.[9] If new CN use cases or solutions are proposed, they can be added to the resources and linked to already available elements of the resource, e.g. the CN dimension used. Finally, feedback is welcome, either via GitHub issues or as mentioned in the "maintenance" section of CNTF.

[6] See the use case: Negotiation of RDF shapes in CNTF.

[7] https://w3id.org/cntf/

[8] https://github.com/YoucTagh/CNTF/blob/master/README.md#citation

[9] https://doi.org/10.5281/zenodo.6504504

## A.4 Summary

Content negotiation is a cornerstone of the Web architecture and a powerful mechanism for selecting the best representation from multiple available alternatives, but over the years, no single resource has collected *everything one needs to know* about it in one place. In this appendix, we introduced CNTF, which is designed to meet a set of requirements extracted from three use cases: (I) Navigable design, (II) Extensible, (III) Categorisable, (IV) Maintainable. We explained how CNTF meets these requirements. We then illustrated the characteristics of CNTF and its relevance to the semantic community.

Appendix B

# Content Negotiation Characteristics illustrations

## B.1 Content Negotiation Dimensions

### Media type

Multiple Listing B.1 shows three representations of the same RDF graph content:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.com/">
  <rdf:Description rdf:about="http://example.com/yousouf">
   <rdf:type rdf:resource="http://example.com/PhD"/>
   <ex:name>Yousouf</ex:name>
  </rdf:Description>
</rdf:RDF>
```

(a) `application/rdf+xml`

```
ex:yousouf a ex:PhD;
  ex:name "Yousouf".
```

(b) `text/turtle`

```
[{ "@id":"http://example.com/PhD"},
 {
  "@id":"http://example.com/yousouf",
  "@type":["http://example.com/PhD"],
  "http://example.com/name":[
    {"@value":"Yousouf"}]
}]
```

(c) `application/ld+json`

Multiple Listing B.1: Snippets of three representations varying over the media type dimension

## Charset

Figure B.1 shows three representations of the same text encoded using different charsets:

```
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   59 6F 75 73 6F 75 66 20 69 73 20 61 20 50 68 44  Yousouf is a PhD
00000010   2E                                               .
```

(a) `UTF-8`

```
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   FF FE 00 00 59 00 00 00 6F 00 00 00 75 00 00 00  .þ..Y...o...u...
00000010   73 00 00 00 6F 00 00 00 75 00 00 00 66 00 00 00  s...o...u...f...
00000020   20 00 00 00 69 00 00 00 73 00 00 00 20 00 00 00   ...i...s... ...
00000030   61 00 00 00 20 00 00 00 50 00 00 00 68 00 00 00  a... ...P...h...
00000040   44 00 00 00 2E 00 0D 00 0A 00                    D.........
```

(b) `UTF-16-BE`

```
           00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000   FF FE 59 00 00 00 6F 00 00 00 75 00 00 00 73 00  .þY...o...u...s.
00000010   00 00 6F 00 00 00 75 00 00 00 66 00 00 00 20 00  ..o...u...f... .
00000020   00 00 69 00 00 00 73 00 00 00 20 00 00 00 61 00  ..i...s... ...a.
00000030   00 00 20 00 00 00 50 00 00 00 68 00 00 00 44 00  .. ...P...h...D.
00000040   00 00 2E 00 00 00 0D 00 0A 00                    ..........
```

(c) `UTF-16-LE`

Figure B.1: Three representations varying over the charset dimension

## Encoding

Multiple Listing B.2 shows four representations of the same content encoded using different encodings:

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
 xmlns:ex="http://example.com/">
  <rdf:Description rdf:about="http://example.com/yousouf">
   <rdf:type rdf:resource="http://example.com/PhD"/>
   <ex:name>Yousouf</ex:name>
  </rdf:Description>
</rdf:RDF>
```

(a) `identity`

```
GzYBAIzTpbbgssSNbkv1kj1ilhmhiaWLAaWQpW4LOX//DYJRBAXBYLMZirBm77auueuz
lZ+H1YU5l1KSChL7ToXWsXP7N6grhTuubBmmH6VKzSVSciwcu0x45JJJSdxxEf+rsH5E
eYyAy/mZ80mfT20A6OfUQcR0Ihy3/zp6HS+4OWQcjcA/jl3zMyBW6vllKRxVHUH9KlgQ
I7oxn5WCZSJW
```

(b) `brotli / base64`

```
H4sIAAAAAAAAA22QOw+DMAyEd35FlM5goEtBPBbUuerWkYKhSCRBSSjh3zc82qFiPPs+
+3RJblhP3ihVJ3hKA8+nBHkl6o63KR11414oyTMnkXUT34srsXauYqtS+tJ6iAGmafKm
sydkC0EUReCHEIaudbhq5ro0Llcn6uwgmh+HpmRDj14lGNDMIWT9UaCqZDdoG4csunyK
UR8ysxiVGJsV3Vg9D7hCEu1GVnjI3V4FhQ1CE/OSYfbYTiXwHSxp4C+OLQH2FrIPDTW7
2jcBAAA=
```

(c) `gzip / base64`

```
bZA7D4MwDIR3fkWUzmCgS0E8FtS56taRgqFIJEFJKOHfNzzaoWI8+z77dEluWE/eKFUn
eEoDz6cEeSXqjrcpHXXjXijJMyeRdRPfiyuxdq5iq1L60nqIAaZp8qazJ2QLQRRF4IcQ
hq51uGrmujQuVyfq7CCaH4emZEOPXiUY0MwhZP1RoKpkN2gbhyy6fIpRHzKzGJUYmxXd
WD0PuEIS7UZWeMjdXgWFDUIT85Jh9thOJfAdLGngL44tAfYWsg8=
```

(d) `deflate / base64`

Multiple Listing B.2: Snippets of four representations varying over the encoding dimension

## Language

Figure B.2 shows three representations of the same content using different languages:

يوسف طالب دكتوراه.

(a) `Arabic`

**Yousouf is a PhD Student.**

(b) `English`

**Yousouf est étudiant en doctorat.**

(c) `French`

Figure B.2: Three textual representations varying over the language dimension

## Feature

Figure B.3 shows four representations of the same content using different features:

(a) `color depth` = 2

(b) `color depth` = 8

(c) `color depth` = 32

(d) `color depth` = 128

Figure B.3: Four image representations varying over the feature dimension

## Capability

Listing B.1 shows parts of the UAProfile of the Samsung `EK-GN120` device:[1]

Listing B.1: Parts of the UAProfile of the Samsung `EK-GN120` device

```
<?xml version="1.0" encoding="UTF-8"?> <rdf:RDF
...
 <prf:CcppAccept>
  <rdf:Bag>
   <rdf:li>application/java-archive</rdf:li>
   <rdf:li>application/xhtml+xml</rdf:li>
   <rdf:li>application/ogg</rdf:li> <rdf:li>text/plain</rdf:li>
   <rdf:li>audio/aac</rdf:li> <rdf:li>audio/mp3</rdf:li>
   <rdf:li>audio/3gpp</rdf:li> <rdf:li>audio/mp4</rdf:li>
   <rdf:li>video/mpeg4</rdf:li> <rdf:li>video/mp4</rdf:li>
   <rdf:li>image/jpeg</rdf:li> <rdf:li>image/jpg</rdf:li>
   <rdf:li>image/png</rdf:li> <rdf:li>text/html</rdf:li>
  ...
 <prf:CcppAccept-Charset>
  <rdf:Bag>
   <rdf:li>ISO-10646-UCS-2</rdf:li> <rdf:li>ISO-8859-1</rdf:li>
   <rdf:li>US-ASCII</rdf:li> <rdf:li>UTF-8</rdf:li>
  </rdf:Bag>
 </prf:CcppAccept-Charset>
 <prf:CcppAccept-Encoding>
  <rdf:Bag>
   <rdf:li>base64</rdf:li>
  </rdf:Bag>
 </prf:CcppAccept-Encoding>
 <prf:CcppAccept-Language>
  <rdf:Seq>
   <rdf:li>en</rdf:li> <rdf:li>ru</rdf:li>
...
```

[1]The full UAProfile of the device: http://wap.samsungmobile.com/uaprof/EK-GN120.xml

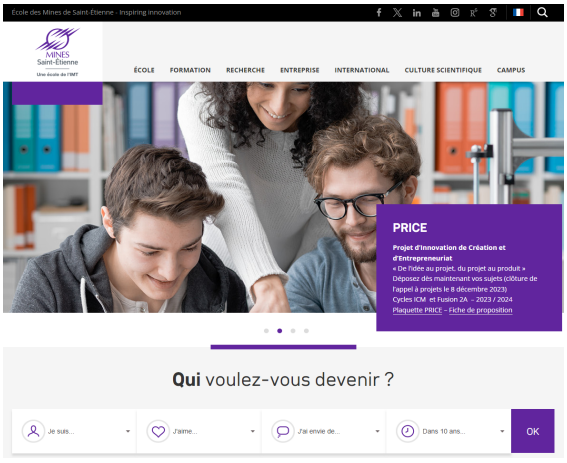
## Time

Figure B.4 shows four representations of the same resource which is École des Mines de Saint-Étienne Website[2] varying over time:[3]


(a) date = `Jan 10, 1997`


(b) date = `Feb 20,1999`


(c) date = `Mar 13, 2012`




(d) date = `Nov 14, 2023`

Figure B.4: Four image representations varying over the time dimension

[2]École des Mines de Saint-Étienne website: http://www.emse.fr/
[3]Web archive: https://web.archive.org/ was used to retrieve old variants.

## Version

Multiple Listing B.3 shows three representations of the same content varying across different versions:

```
{
 "name":"YoucTagh",
 "title":"PhD"
}
```

(a) `Version` = 1

```
{
 "fullname":"YoucTagh",
 "title":"PhD"
}
```

(b) `Version` = 2

```
{
 "firstname":"Youc",
 "lastname":"Tagh",
 "degree":"PhD"
}
```

(c) `Version` = 3

Multiple Listing B.3: Three representations varying over the version dimension

## CRS

Multiple Listing B.4 shows two representations of the position of École des Mines de Saint-Étienne building varying across two coordinate reference systems:

```
{
 "crs": {
  "type": "link",
  "properties": {
   "href": "https://spatialreference.org/ref/epsg/4326/json/",
   "type": "json"}},
 "type": "FeatureCollection",
 "features": [{
  "type": "Feature",
  "properties": {},
  "geometry": {
   "coordinates": [4.4083336786915766,45.422534040839395],
   "type": "Point"
}}]}
```

(a) `EPSG 4326`

```
{
 "crs": {
  "type": "link",
  "properties": {
   "href": "https://spatialreference.org/ref/epsg/2154/json/",
   "type": "json"}},
 "type": "FeatureCollection",
 "features": [{
  "type": "Feature",
  "properties": {},
  "geometry": {
   "coordinates": [810134.4393036033,6481319.505495705],
   "type": "Point"
}}]}
```

(b) `EPSG 2154`

Multiple Listing B.4: Two representations varying over the CRS dimension

## Formatting

Multiple Listing B.5 shows three representations of a citation using different formatting:

```
Frank, H. S. (1970). The Structure of Ordinary Water. Science,
   169(3946), 635-641. https://doi.org/10.1126/science.169.3946.635
```

(a) `text/x-bibliography; style=apa`

```
Frank, H.S. (1970). The Structure of Ordinary Water. Science 169,
   635-641.
```

(b) `text/x-bibliography; style=cell`

```
@article{1970,
 title={The Structure of Ordinary Water},
 volume={169},
 ISSN={1095-9203},
 url={http://dx.doi.org/10.1126/science.169.3946.635},
 DOI={10.1126/science.169.3946.635},
 number={3946},
 journal={Science},
 publisher={American Association for the Advancement of Science (
    AAAS)},
 author={Frank, Henry S.},
 year={1970}, month={Aug},
 pages={635-641}
}
```

(c) `text/x-bibliography; style=bibtex`

Multiple Listing B.5: Two representations varying over the Formatting dimension

## Presentation

Let say we have the following input:

Listing B.2: RDF presentation input example

```
I10:::2023/02/18:::Sousou->Kheiro:::Reminder:::Don't forget tomato!
I21:::2023/05/18:::Wich->Moh:::Update:::We now leave at ten
```

If we negotiate for the presentation with the following lifting rule:

Listing B.3: RDF presentation lifting rule example

```
...
GENERATE {
 ?noteIRI a mail:note; mail:from ?from; mail:to ?to;
  mail:heading ?heading; mail:body ?body.
 ?dmy a ex:DateTime; ex:day ?gDay; ex:month ?gMonth; ex:year ?gYear.
 ?noteIRI mail:when ?dmyIRI .
}
ITERATOR iter:Split(?message, "\n") AS ?note
WHERE {
 BIND(fn:SplitAtPosition(?note,":::", 0) AS ?noteId)
 BIND(fn:SplitAtPosition(?note,":::", 1) AS ?dmy)
 BIND(fn:SplitAtPosition(?dmy,"/", 0) AS ?year)
 BIND(fn:SplitAtPosition(?dmy,"/", 1) AS ?month)
 BIND(fn:SplitAtPosition(?dmy,"/", 2) AS ?day)
 BIND(STRDT(?year,xsd:gYear) as ?gYear)
 BIND(STRDT(?month,xsd:gMonth) as ?gMonth)
 BIND(STRDT(?day,xsd:gDay) as ?gDay)
 BIND(fn:SplitAtPosition(?note,":::", 2) AS ?tofrom)
 BIND(fn:SplitAtPosition(?tofrom,"->", 0) AS ?to)
 BIND(fn:SplitAtPosition(?tofrom,"->", 1) AS ?from)
 BIND(fn:SplitAtPosition(?note,":::", 3) AS ?heading)
 BIND(fn:SplitAtPosition(?note,":::", 4) AS ?body)
 BIND(IRI(CONCAT("http://example.com/note/",?noteId)) as ?noteIRI)
 BIND(IRI(CONCAT("http://example.com/DateTime/",?year,"/",?month
    ,"/",?day)) as ?dmyIRI)
}
```

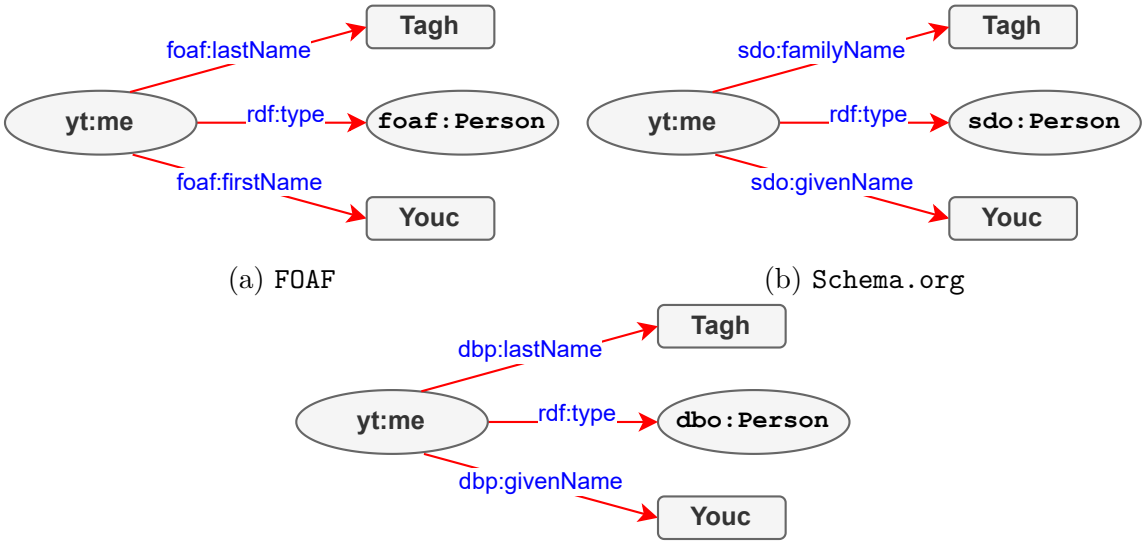
We obtain the following result:

Listing B.4: RDF presentation output of negotiating

```
...
node:I10 a mail:note ;
 mail:when <http://example.com/DateTime/2023/02/18> ;
 mail:to "Sousou";  mail:heading "Reminder" ;
 mail:from "Kheiro"; mail:body "Don't forget tomato!" .
<http://example.com/DateTime/2023/02/18> a ex:DateTime ;
 ex:year "2023"^^xsd:gYear; ex:month "02"^^xsd:gMonth ;
 ex:day "18"^^xsd:gDay .
node:I21 a mail:note ;
 mail:when <://example.com/DateTime/2023/05/18> ;
 mail:to "Wich"; mail:heading "Update" ;
 mail:from "Moh"; mail:body "We now leave at ten" .
<http://example.com/DateTime/2023/05/18> a ex:DateTime;
 ex:year "2023"^^xsd:gYear; ex:month "05"^^xsd:gMonth;
 ex:day "18"^^xsd:gDay .
```

## Vocabulary

Figure B.5 shows three RDF representations varying over the vocabulary dimension:




(a) `FOAF` (b) `Schema.org`

(c) `DBpedia`

Figure B.5: Three RDF representations varying over the vocabulary dimension

## Schema

Multiple Listing B.6 shows a valid and an invalid XML representations varying over the schema dimension:

```
<person>
 <name>YoucTagh</name>
 <title>PhD</title>
</person>
```

(a) Valid

```
<person>
 <fullname>YoucTagh</fullname>
 <title>PhD</title>
</person>
```

(b) Invalid

```
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:element name="person">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="name" type="xs:string"/>
   <xs:element name="title" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

(c) Negotiated XML Schema

Multiple Listing B.6: A valid and an invalid XML representations varying over the schema dimension

## Profile

Throughout the dissertation we have presented multiple instances of profile negotiation, see for example Listing 5.1, Listing 5.3 and Listing 5.4.

## Overview of the CN Dimension Vocabulary

The following vocabulary complements *CNCO* the ontology presented in Section 8.3:

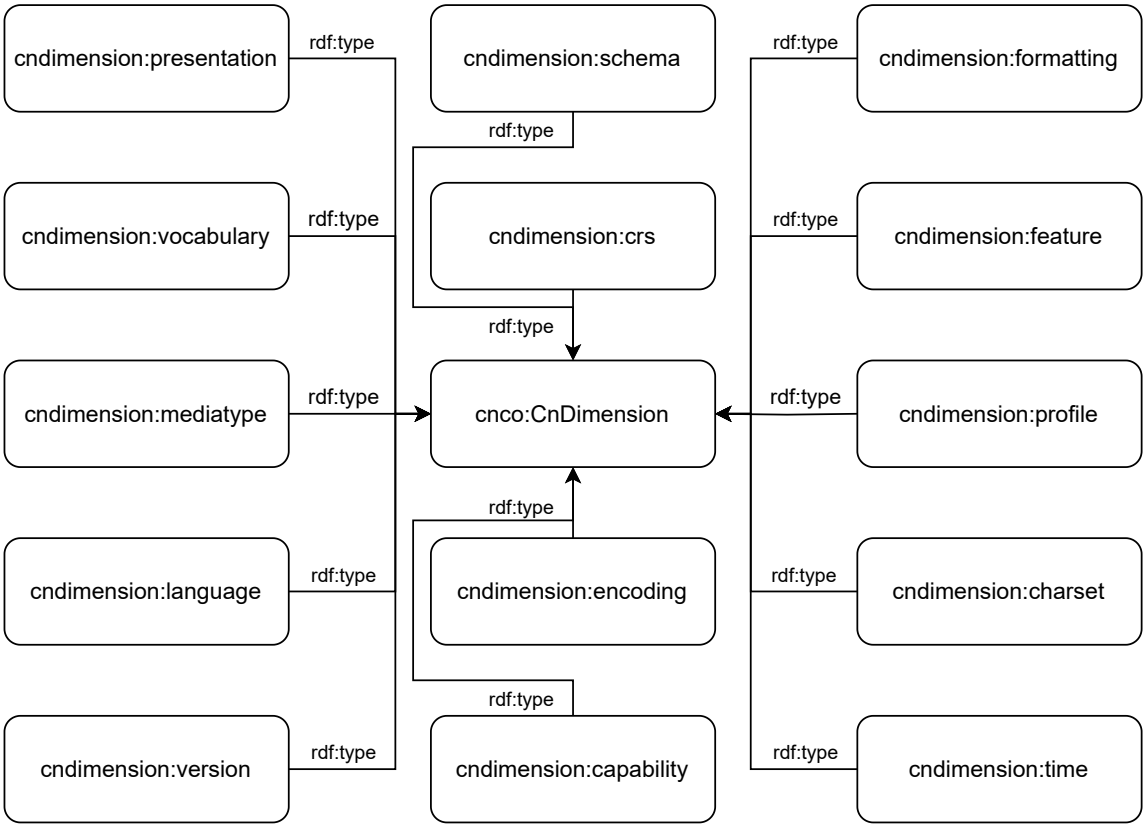



Figure B.6: Overview of the content negotiation dimension vocabulary

## B.2 Content Negotiation Styles

### Proactive

A client request:

Listing B.5: A client request for a resource using content negotiation with proactive style

```
1 GET /species/abies_numidica HTTP/1.1
2 Host: youctagh.com
3 accept: text/plain
```

A server response:

Listing B.6: A server response to a request using content negotiation with proactive style

```
HTTP/1.1 200 OK
Date: Mon, 10 Nov 2023 12:38:39 GMT
Content-Location: abies_numidica.txt
Content-Type: text/plain
Vary: accept
Content-Length: 80

Abies numidica, the Algerian fir, is a species of fir found only
   in Algeria, ...
```

## Reactive

A client request:

Listing B.7: A client request for a resource using content negotiation with reactive style

```
GET /species/abies_numidica HTTP/1.1
Host: youctagh.com
```

A server response:

Listing B.8: A server response to a request using content negotiation with reactive style

```
HTTP/1.1 300 Multiple Choices
Date: Mon, 10 Nov 2023 12:38:39 GMT
Link:
    <http://youctagh.com/species/abies_numidica?format=txt>;
        rel="canonical";
        type="text/plain",
    <http://youctagh.com/species/abies_numidica?format=json>;
        rel="alternate";
        type="application/json",
    <http://youctagh.com/species/abies_numidica?format=xml>;
        rel="alternate";
        type="application/xml"
Content-Type: text/html

<!DOCTYPE html>
...
 <h1>Available Representations</h1>
  <table>
   <tr> <th>Representations</td>
        <th>Media type</td>
        <th>isDefault</td> </tr>
```

```
22      <tr>
23       <td><a href="http://youctagh.com/species/abies_numidica?format
            =txt">Representation 1</a></td>
24       <td><span>text/plain</span></td>
25       <td><span>Yes</span></td>
26      </tr>
27  ...
```

In this example the user agent generates an HTML page of the available representations for the user to choose from.

# Available Representations

| Representations | Media type | isDefault |
|---|---|---|
| Representation 1 | text/plain | Yes |
| Representation 2 | application/json | No |
| Representation 3 | application/xml | No |

Figure B.7: An HTML page of the available representations for the user to choose from generated by the user agent

### Request

A response from a server to a request. In the same request, negotiating for `application/json` for subsequent requests:

Listing B.9: A server response negotiating a specific media type using the request content negotiation style

```
HTTP/1.1 200 OK
Date: Mon, 10 Nov 2023 12:38:39 GMT
Content-Location: abies_numidica.txt
Content-Type: text/plain
Vary: accept
Allow: GET, PUT, POST, OPTIONS, HEAD, DELETE, PATCH
Accept-Patch: application/json
Content-Length: 80

Abies numidica, the Algerian fir, is a species of fir found only
   in Algeria, ...
```

A client `POST` request that uses the negotiated media type:

Listing B.10: A client POST request with the requested media type

```
POST /species HTTP/1.1
Host: youctagh.com
Allow: GET, PUT, POST, HEAD
Content-Type: application/json
Content-Length: 204
Slug: abies_cilicica

{
 "name":"Abies cilicica",
 ...
}
```

## Transparent

A client request:

Listing B.11: A client request for a resource using content negotiation with transparent style

```
GET /species/abies_numidica HTTP/1.1
Host: youctagh.com
accept: text/plain
```

Note that the request is similar to the one sent in proactive style. However, as part of the response we can see `TCN` as well as the additional `negotiate` in the `Vary` header:

Listing B.12: A response to a request using content negotiation with transparent style

```
HTTP/1.1 200 OK
Date: Mon, 10 Nov 2023 12:38:39 GMT
Content-Location: abies_numidica.txt
Content-Type: text/plain
Vary: accept, negotiate
TCN: choice
Content-Length: 80

Abies numidica, the Algerian fir, is a species of fir found only
   in Algeria, ...
```

## Conditional

A client request:

Listing B.13: A client request for a resource using content negotiation with conditional style

```
GET /species/abies_numidica HTTP/1.1
Host: youctagh.com
```

Note that the request is similar to the one sent in reactive style. However, in the response we can see that all the representations are being returned separated by a boundary:

Listing B.14: A response to a request using content negotiation with conditional style

```
HTTP/1.1 200 OK
Date: Mon, 10 Nov 2023 12:38:39 GMT
Content-Type: multipart/alternative; boundary=SEPARATION
--SEPARATION
Content-Type: application/xml
<root>
 <name>Abies numidica</name>
 <country>Algeria</country>
</root>
--SEPARATION
Content-Type: application/json
{
 "name" : "Abies numidica",
 "country" : "Algeria"
}
--SEPARATION
Content-Type: text/plain
```

```
18 Abies numidica, the Algerian fir, is a species of fir found only
      in Algeria, ...
19 --SEPARATION--
```

### Active

A client request:

Listing B.15: A client request for a resource using content negotiation with active style

```
1 GET /species/abies_numidica HTTP/1.1
2 Host: youctagh.com
3 accept: text/plain
```

Note that the request is similar to some previous ones. However, in the response we would have some executable code that would issue additional specific requests:

Listing B.16: A JavaScript code contained in a response to a request using content negotiation with active style

```
1  var userLang = navigator.language || navigator.userLanguage;
2  getRepresentation(userLang);
3
4  async function getRepresentation(userLang) {
5   const response = await fetch("http://localhost/cn-test/1/yousouf"
6    ,{headers: { "accept-language": userLang }}
7   );
8   response.text().then((text)=>{
9    console.log(text);
10  })
11 }
```

### Overview of the CN Style Vocabulary

The following vocabulary complements *CNCO* the ontology presented in Section 8.3:

## B.3 Content Negotiation Constraint Conveyance Means

In this section, we present some client requests using different constraint conveyance means when content negotiation is used:

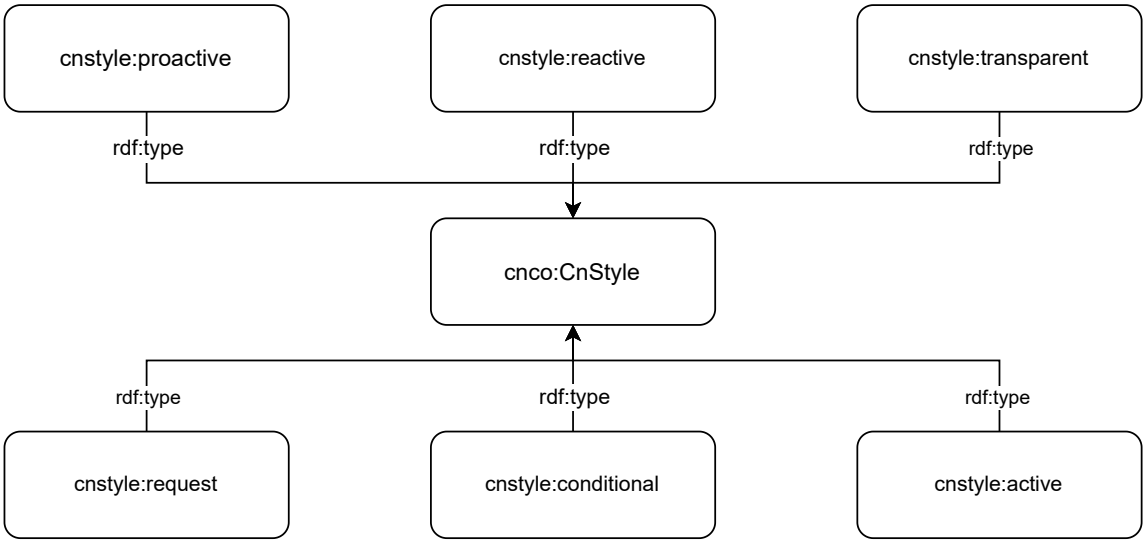



Figure B.8: Overview of the content negotiation styles vocabulary

## Header-Based

Listing B.17: An HTTP client request with header-based constraint conveyance means

```
1 GET /species/abies_numidica HTTP/1.1
2 Host: youctagh.com
3 accept: text/plain
```

## URI-Based – Path Extension

Listing B.18: An HTTP client request with URI-based path extension constraint conveyance means

```
1 GET /species/abies_numidica.txt HTTP/1.1
2 Host: youctagh.com
```

## URI-Based – ARK

Listing B.19: An HTTP client request with URI-based ARK constraint conveyance means

```
1 GET /ark:12345/yt1sp/abies.txt HTTP/1.1
2 Host: youctagh.com
```

## URI-Based – QSA

Listing B.20: An HTTP client request with URI-based QSA constraint conveyance means

```
1 GET /species/abies_numidica?format=txt HTTP/1.1
2 Host: youctagh.com
```

## B.4 Content Negotiation Protocols

In this section, we present some requests and responses using different protocols when content negotiation is used:

### HTTP request/response

A client request:

Listing B.21: An HTTP client request using content negotiation

```
1 GET /species/abies_numidica HTTP/1.1
2 Host: youctagh.com
3 accept: text/plain
```

A server response:

Listing B.22: An HTTP server response to a request using content negotiation

```
1 HTTP/1.1 200 OK
2 Date: Mon, 10 Nov 2023 12:38:39 GMT
3 Content-Location: abies_numidica.txt
4 Content-Type: text/plain
5 Vary: accept
6 Content-Length: 80
7
8 Abies numidica, the Algerian fir, is a species of fir found only
     in Algeria, ...
```

### CoAP request/response

A client request:

Listing B.23: A CaAP client request using content negotiation

```
1 Version: 1
2 Type: Confirmable (0)
3 Token Length: 0
4 Code: GET (1)
5 Message ID: 29545
```

```
6  Opt Name: #1: Uri-Path: species/abies_numidica
7  Opt Name: #2: Accept: text/plain; charset-utf-8 (0)
```

A server response:

Listing B.24: A CoAP server response to a request using content negotiation

```
1  Version: 1
2  Type: Achnowledgment (2)
3  Token Length: 0
4  Code: 2.05 Content (69)
5  Message ID: 29545
6  Opt Name: #1: Content-Format: text/plain; charset-utf-8 (0)
7  Payload: "Abies numidica, the Algerian fir, is a species of fir
       found only in Algeria, ..."
```

## Overview of the CN Protocol Vocabulary

The following vocabulary complements *CNCO* the ontology presented in Section 8.3:

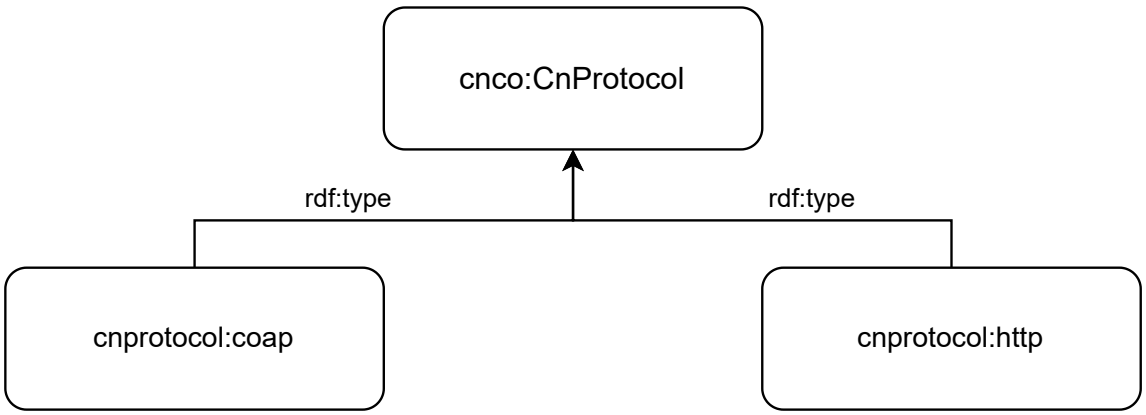



Figure B.9: Overview of the content negotiation protocol vocabulary

Appendix C

# Résumé des points clés de la thèse

## C.1 Introduction générale

En 1989, Tim Berners-Lee a proposé un nouveau système de gestion de l'information, le "World Wide Web", que nous appelons communément "le Web". Il fournit un accès universel à un vaste ensemble de ressources interconnectées [13]. Une ressource peut être un arbre, une personne ou n'importe quoi d'autre. En outre, chaque ressource peut avoir différentes représentations, c'est-à-dire des documents Web, qui reflètent son état. Prenons l'exemple de l'espèce d'arbre *Abies numidica*. Elle peut avoir différentes représentations qui varient selon plusieurs dimensions : le type de média tel que HTML ou PDF ; ou la langue tel que l'anglais, le français ou l'arabe. La négociation de contenu est le mécanisme utilisé pour sélectionner une représentation appropriée parmi celles disponibles. En 2001, Tim Berners-Lee, James Hendler et Ora Lassila ont publié un article dans le magazine Scientific American intitulé "The Semantic Web" [21]. L'article décrit le Web Sémantique comme une extension du Web qui donne une structure et un sens au contenu disponible dans cet espace. En conséquence, nous avons assisté à une évolution d'un Web de documents vers un Web de données.

Les efforts de standardisation déployés par l'IETF et l'W3C ont contribué à faire de cette vision une réalité : L'identification, la localisation et l'accès aux ressources sur Internet sont désormais bien définis grâce à URI [18]. HTTP a été développé et affiné pour un système d'information hypermédia distribué et collaboratif [58, 59, 61]. Les principaux éléments de conception du Web, ainsi que les bonnes pratiques pour les principes et les propriétés qu'ils soutiennent, sont examinés dans la spécification de l'architecture du World Wide Web [88]. Les relations entre les ressources sur le Web et leur type sont définies dans le document "Web Linking" [129].

Des efforts de standardisation similaires ont été déployés dans le domaine du Web Sémantique. Les propositions initiales portaient sur RDF [48, 82] et RDFS [32], qui permettent la description des ressources. OWL et sa version la plus récente OWL 2 [73] ont été proposées pour permettre la définition et la modélisation de connaissances plus complexes et plus riches. En outre, la nécessité d'un langage

permettant la validation des données RDF par rapport à un ensemble de contraintes a motivé le développement de SHACL [97] et de ShEx [137].

En conséquence, la négociation de contenu peut évoluer ; le client ne négocie plus seulement le type de média et la langue, mais aussi la structure et la sémantique. Dès 2006, des discussions sur les attentes en matière de vocabulaire et la négociation sémantique de contenu ont eu lieu sur la liste de diffusion Semantic Web.[1] En 2017, le W3C a créé un groupe de travail (DXWG), dont l'un des objectifs est d'étudier la négociation de contenu dans une nouvelle dimension appelée *profil* [183]. Dans ce contexte, l'objectif de cette thèse est : (1) d'étudier la négociation de contenu, et plus particulièrement la négociation de contenu par profil ; (2) de plaider en faveur d'une négociation de contenu fine et souple dans cette dimension, par exemple pour définir des contraintes à grain fin ; et de traiter le cas du non-respect des contraintes optionnelles. Tout cela en tenant compte du fait que la négociation de contenu n'est pas un processus monolithique, mais consiste en plusieurs étapes : découverte, formulation de la demande, sélection et/ou adaptation, indication de la réponse, interprétation de la réponse.

## C.2 Motivation et limites identifiées

Dès le début, le modèle du Web a permis de négocier le format des données entre les fournisseurs de contenu (serveurs) et les demandeurs (clients), principalement pour résoudre les problèmes d'interopérabilité, comme le décrit le document "World-WideWeb - Executive Summary" [14]. Cette vision de la négociation de contenu se retrouve dans les premières versions de HTTP [15]. La négociation sur la langue, le charset et le encodage a été introduite peu après [19, Section D.2]. Au cours de la transition du Web de documents au Web de données, la négociation de contenu a également évolué. Toutefois, il manque un modèle abstrait du problème de la négociation de contenu qui rende compte du processus de négociation dans le Web de documents lorsque la négociation porte sur des dimensions telles que le type de média et la langue. Un tel modèle devrait également être extensible aux pratiques actuelles de négociation de contenu sur le Web Sémantique. Il s'agit d'une limitation sérieuse si nous voulons formaliser la négociation de contenu fine et souple envisagée.

**Limitation 1.** Il est nécessaire d'adopter une formalisation qui tienne compte de l'évolution du Web des documents et du Web des données.

Lors de la négociation de contenu dans HTTP, les valeurs de qualité pourraient

[1] https://lists.w3.org/Archives/Public/semantic-web/2006Jul/

être utilisées pour indiquer le poids des différentes contraintes [60, Section 3.9]. Cette technique est également utilisée dans les dimensions ultérieures de la négociation de contenu telles que *profil* [161, 163]. Toutefois, dans une telle dimension, les contraintes pourraient être plus fines. Par exemple, un profil pourrait être un graphe de formes SHACL et chaque forme pourrait définir plusieurs contraintes. Pourtant, le poids est appliqué à l'ensemble du profil. De même, le serveur, lorsqu'il traite la demande, effectue la validation ou répond, pourrait également avoir besoin de techniques plus fines pour communiquer les résultats de la négociation de contenu, car les codes d'état et les en-têtes de réponse HTTP existants ne sont pas toujours suffisants. En outre, un client peut avoir des contraintes facultatives et souhaiter les communiquer. Par exemple, une forme SHACL définissant trois contraintes pour trois propriétés, la première étant facultative, les deux autres obligatoires. Cette sémantique n'est ni spécifiée ni traitée dans le processus de négociation de contenu. La méthode "tout ou rien" de traitement des demandes prévaut.

**Limitation 2.** La formulation de la demande et le traitement des contraintes ne sont pas suffisamment fins et souples, pas plus que le rapport de validation et l'interprétation de la réponse.

Au fil des ans, la négociation de contenu a été utilisée dans divers cas d'utilisation allant des fonctionnalités des appareils [36], de l'archivage [91], au formatage des citations [198]. De même, la négociation de la structure et de la sémantique dans la dimension du profil est un besoin pratique dans de multiples cas d'utilisation [140]. Cependant, même si différents cas d'utilisation peuvent adopter les mêmes styles de négociation de contenu [61, Section 12], la mise en œuvre réelle peut différer dans l'utilisation de modèles plus fins. Par conséquent, la collecte et l'étude des différents modèles plus fins sont essentielles pour que la dimension de profil soit acceptée et fournisse des solutions générales et réutilisables.

**Limitation 3.** Il n'existe pas de typographie pour les modèles plus fins de négociation de contenu dans la dimension du profil.

Les différents fournisseurs de représentations utilisent des identifiants différents pour la même ressource. Par conséquent, un client qui négocie la représentation d'une ressource avec l'un d'entre eux ne prend en compte qu'un sous-ensemble de toutes les représentations disponibles. Cela signifie que si aucune représentation ne satisfait les contraintes du client sur le serveur actuel, une représentation alternative sur un autre serveur peut y répondre.

**Limitation 4.** Il n'existe aucun moyen de négocier des représentations distribuées sur plusieurs serveurs si nécessaire.

Le Web n'est pas utilisé uniquement par des humains, mais aussi par des agents logiciels [25]. En outre, les utilisateurs finaux interagissent avec les serveurs par l'intermédiaire d'agents utilisateurs tels que les navigateurs. L'implication initiale de ces agents dans le processus de négociation de contenu s'est faite par l'envoi d'en-têtes préconfigurés pour un charset et un encodage particuliers, ou un langage que l'utilisateur final peut comprendre. Toutefois, ces agents logiciels n'ont toujours pas la capacité de localiser les ressources négociables et de s'engager ensuite de manière autonome dans la négociation de contenu.

**Limitation 5.** Les agents logiciels sont limités dans leur capacité à engager une négociation de contenu de manière autonome.

Notre thèse est que nous pouvons remédier aux limitations susmentionnées en proposant tout d'abord un modèle abstrait pour la négociation de contenu qui reflète le comportement classique de la négociation de contenu dans ses dimensions d'origine, puis en l'étendant pour saisir les exigences d'un contexte de Web sémantique. Ensuite, en utilisant les travaux déjà disponibles dans l'espace du Web Sémantique, tels que les liens d'équivalence et la validation sémantique, pour permettre une négociation de contenu plus fin et plus souple qui adhère à des modèles plus fins bien définis. Enfin, doter les agents logiciels de la capacité de s'engager dans la négociation de contenu de manière autonome si nécessaire.

## C.3 Questions de recherche abordées

L'objectif principal de cette thèse est de contribuer au développement d'un cadre de négociation de contenu sémantique dans lequel les agents web s'engagent dans une négociation de contenu plus fine et plus souple. Notre approche s'appuie sur les résultats les plus récents de la recherche sur le Web sémantique et les systèmes multi-agents. Pour mettre en place ce cadre, nous abordons cinq questions de recherche, que nous présentons dans les paragraphes suivants. Chaque question est axée sur la résolution d'une des limitations identifiées.

Un élément clé de notre thèse est la formalisation du problème de la négociation de contenu. La négociation de contenu existe depuis plusieurs décennies et a été utilisée dans différents scénarios et contextes, ce qui soulève le défi de choisir le bon niveau d'abstraction, ainsi que le choix des éléments à inclure dans le modèle

abstrait de manière à le rendre inclusif et extensible pour une utilisation future (cf. Limitation 1). À cette fin, nous introduisons et formalisons des concepts qui reflètent la pratique actuelle et nous montrons comment ils s'intègrent dans la manière dont le protocole HTTP spécifie la négociation de contenu. Nous étendons ensuite le modèle général dans le contexte du Web Sémantique pour permettre la négociation de représentations conformes à certains profils.

**Questions de recherche 1.** Comment formaliser le problème de la négociation de contenu d'une manière générale et extensible afin de prendre en compte la négociation par profil ?

Un autre élément clé de notre thèse est l'assouplissement de la négociation de contenu. Par exemple, pour être en mesure de définir des contraintes fines et de gérer le cas du non-respect des contraintes optionnelles. Cependant, cela doit être fait en plusieurs étapes, puisque la négociation de contenu consiste en plusieurs étapes (cf. Limitation 2). À cette fin, nous proposons des solutions permettant au client de spécifier des contraintes plus fines grâce à l'utilisation de sévérités de contraintes. Ensuite, du côté du serveur, nous proposons d'assouplir la validation en tenant compte de ces contraintes plus fines pour sélectionner les représentations adéquates et communiquer les résultats de la validation au client, qui peut à son tour interpréter et utiliser ces réponses.

**Questions de recherche 2.** Comment un client et un serveur peuvent-ils engager dans un processus de négociation de contenu souple ?

Un élément clé de notre thèse est l'identification, la collecte et la classification de modèles plus fins utilisés lors de la négociation de contenu dans la dimension du profil (cf. Limitation 3). À cette fin, nous introduisons la notion de *modèle plus fin de négociation de contenu*, qui renvoie à des modèles répondant à des questions plus précises que les styles de négociation de contenu, telles que : le serveur utilise-t-il une vérification de conformité stricte ou souple ? Le contrôle de conformité est-il effectué à la volée ou est-il prétraité ? La validation est-elle effectuée localement par le serveur ou par un tiers ? Nous montrons comment certains de ces modèles plus fins de négociation de contenu peuvent tirer parti des standards et de l'infrastructure web actuels.

**Questions de recherche 3.** Quels sont les modèles les plus fins qui peuvent façonner le processus de négociation dans la dimension du profil ?

Sur le Web, une ressource peut avoir différents identifiants. Par conséquent, les représentations des ressources peuvent être réparties sur plusieurs serveurs, alors que les scénarios habituels de négociation de contenu ne prennent en compte qu'un seul serveur (cf. Limitation 4). À cette fin, nous proposons d'utiliser des liens d'équivalence pour découvrir des représentations potentiellement acceptables et par la suite adapter l'architecture de négociation de contenu en conséquence.

**Questions de recherche 4.** Comment négocier une variante appropriée d'une ressource lorsque ses représentations sont distribuées ?

Les clients web, en particulier les agents logiciels, n'ont pas toujours une connaissance a priori des caractéristiques de négociation de contenu utilisées dans un serveur web, par exemple les styles de négociation de contenu utilisés dans la négociation, ou les moyens préférés pour transmettre les contraintes. Même lorsqu'une documentation est disponible, elle est généralement conçue pour les humains. Un exemple de ce type de documentation est une API Swagger décrivant les attentes en matière de négociation de contenu. Par conséquent, il est difficile pour les agents logiciels de s'engager dans la négociation de contenu avec les serveurs web (cf. Limitation 5). À cette fin, nous proposons l'ontologie de la négociation de contenu pour capturer les connaissances sur les propriétés de négociation de contenu des serveurs web. Nous illustrons ensuite comment les recherches récentes sur les systèmes multi-agents peuvent être utilisées conformément à l'ontologie proposée pour permettre aux agents autonomes de s'engager dans la négociation sémantique de contenu.

**Questions de recherche 5.** Comment permettre à des agents autonomes de découvrir et d'exploiter des ressources négociables ?

## C.4 Structure et plan de la thèse

Cette thèse est structurée en quatre parties :

Dans la Partie I, nous analysons l'état de l'art pour mettre en évidence le début et l'évolution de la négociation de contenu sur le Web et les travaux et technologies connexes qui peuvent être utilisés pour mettre en place le cadre de négociation de contenu sémantique envisagé.

Dans le Chapitre 2, nous fournissons les connaissances de base nécessaires pour le reste de la thèse. Nous commençons par présenter le problème de la négociation

de contenu à la lumière de l'émergence du Web. Nous passons ensuite en revue le Web Sémantique et ses composants utilisés tout au long de la thèse.

Dans le Chapitre 3, nous passons d'abord en revue les différentes étapes de la négociation de contenu ainsi que la manière dont les travaux antérieurs ont tenté de la formaliser. Nous examinons ensuite les différentes caractéristiques de la négociation de contenu : dimensions, styles et moyens de transmission des contraintes. Ensuite, nous examinons de plus près la dimension du profil, en nous concentrant sur deux questions principales : comment un client peut-il spécifier un profil ? et comment les préférences peuvent-elles être transmises dans cette dimension de la négociation de contenu ?

Dans la deuxième partie, nous présentons nos contributions à la conception d'une négociation de contenu plus fine et plus souple. Dans le Chapitre 4, nous abordons la Question de recherche 1. Nous présentons notre proposition de modèle général pour la négociation de contenu à travers un cas d'utilisation, en commençant par le cas qui reflète les pratiques actuelles, et nous discutons de la manière dont il s'intègre dans la façon dont HTTP spécifie la négociation de contenu. Nous montrons ensuite comment nous avons étendu le modèle général dans le contexte du Web Sémantique pour permettre la négociation de contenu dans la dimension du profil.

Dans le Chapitre 5, nous abordons la Question de recherche 2. Nous décrivons comment nous améliorons le processus de négociation de contenu en utilisant la validation sémantique tout en étant guidés par les phases de négociation de contenu : d'abord en indiquant des contraintes plus fines, puis en assouplissant la validation et en signalant une validation plus fine, et enfin en interprétant le rapport de validation.

Dans le Chapitre 6, nous abordons la Question de recherche 3. Nous discutons de modèles plus fins possibles pour gérer la négociation de contenu dans la dimension du profil. En particulier, nous montrons comment certains d'entre eux peuvent utiliser les normes et l'infrastructure Web actuelles, et nous les illustrons à l'aide d'un cas d'utilisation et des efforts actuels des groupes DXWG et IETF.

Dans la troisième partie, nous présentons l'évaluation de notre travail. Dans le Chapitre 7, nous abordons la Question de recherche 4. Tout d'abord, nous examinons plus en détail le problème de la distribution des représentations. Ensuite, nous présentons une approche pour effectuer la négociation de contenu lorsque les représentations sont distribuées en utilisant des liens d'équivalence avec vérification de la conformité à la volée. À cette fin, nous fournissons deux algorithmes : Le premier dans un contexte de base (c'est-à-dire en considérant uniquement les contraintes de type de média) et le second dans un contexte sémantique (c'est-à-dire en considérant les profils). Une mise en œuvre des algorithmes ainsi que des expériences

distinctes ont été menées pour mesurer les avantages et évaluer les besoins en temps de ces méthodes.

Dans le Chapitre 8, nous abordons la Question de recherche 5. Nous commençons par donner un aperçu des concepts issus de la recherche sur les systèmes multi-agents qui seraient utiles pour permettre la négociation de contenu sémantique dans ces systèmes. Ensuite, nous discutons de notre contribution à la facilitation de l'interaction de systèmes hétérogènes. Tout d'abord, nous décrivons un cas d'utilisation de systèmes multi-agents basés sur le Web où la négociation sémantique de contenu est nécessaire. Ensuite, nous présentons l'ontologie de négociation de contenu. Enfin, nous illustrons les étapes de la mise en place et de l'implémentation d'une telle idée.

Dans la Partie IV, nous présentons un résumé de nos travaux et soulignons les directions à suivre pour les recherches futures.

## C.5 Revue des principales contributions

L'objectif de cette thèse est de contribuer au développement d'un cadre de négociation de contenu sémantique dans lequel les agents web s'engagent dans une négociation de contenu plus *fine* et *souple*, en tenant compte du fait que ce processus n'est pas monolithique, mais consiste en plusieurs phases : (1) découverte ; (2) formulation de la demande ; (3) sélection et/ou adaptation ; (4) indication de la réponse ; (5) interprétation de la réponse.

Nous avons proposé d'atteindre cet objectif en utilisant des technologies et des outils sémantiques pour améliorer la négociation de contenu sous plusieurs aspects : (1) découvrir des représentations potentielles grâce à des liens d'équivalence, en plus de découvrir des ressources négociables grâce à leurs descriptions, qui utilisent l'ontologie de négociation de contenu que nous avons proposée ; (2) indiquer des contraintes plus fines grâce à la sévérité et savoir comment engager dans la négociation de contenu grâce aux descriptions des ressources négociables ; (3) assouplir la validation ; (4) rapporter une validation plus fine ; (5) interpréter le rapport de validation et réagir à la réponse en fonction des descriptions des ressources négociables.

Dans ce qui suit, nous présentons un résumé de nos contributions et discutons des perspectives de recherche future. Nous résumons et organisons nos contributions en fonction des cinq questions de recherche que nous abordons dans cette thèse :

Pour la Question de recherche 1, nous avons proposé CON-NEG, un modèle abstrait général pour la négociation de contenu (voir Section 4.2.1), où nous avons choisi le bon niveau d'abstraction et les éléments à inclure dans le modèle abstrait d'une

manière qui le rend inclusif et extensible pour une utilisation future. Nous avons démontré l'extensibilité de notre modèle à un contexte sémantique pour permettre la négociation de représentations conformes à certains profils (voir Section 4.2.2). Nous avons proposé S-CON-NEG, un modèle pour une négociation de contenu plus fine et plus souple (voir Section 5.3). Nous avons discuté de l'applicabilité de nos modèles CON-NEG et S-CON-NEG dans les sections 4.3 et Section 4.3 respectivement. Des parties de cette proposition ont été présentées à la 20ème Rencontre des Jeunes Chercheurs en Intelligence Artificielle (RJCIA 2022) [168].

Pour la Question de recherche 2, nous avons proposé une approche pour améliorer le processus de négociation de contenu en le rendant plus fin et plus souple, tout en étant guidé par les étapes de négociation de contenu et en suivant les principes architecturaux généraux du Web (voir Section 5.5). À cette fin, nous avons fourni des solutions pour spécifier des contraintes plus fines grâce à l'utilisation de contraintes assouplies. Ensuite, du côté du serveur, nous avons proposé d'assouplir la validation en prenant en compte ces contraintes plus fines et en utilisant des méthodes de notation pour sélectionner les représentations appropriées et communiquer les résultats de la validation au client, qui peut à son tour interpréter et utiliser ces réponses améliorées (voir Section 5.2). Nous avons démontré l'applicabilité de notre approche dans la section 7.4.2, où un client est en mesure de demander des contraintes fines et un serveur les prend en compte lors de la sélection de la meilleure option disponible, en particulier dans le cas de la non-conformité des contraintes optionnelles.

Des parties de cette proposition ont été présentées à la 23e Conférence internationale sur l'ingénierie et la gestion des connaissances (EKAW 2022) [166], et à la 20e Rencontre des jeunes chercheurs en intelligence artificielle (RJCIA 2022) [168].

Pour la Question de recherche 3, nous avons identifié, collecté et classé des modèles plus fins utilisés lors de la négociation de contenu dans la dimension du profil (voir Section 6.1). Ces modèles plus fins répondent à des questions plus précises que les styles de négociation de contenu, en particulier lorsqu'une négociation de contenu plus fine et plus souple est adoptée. Nous avons illustré la manière dont ces modèles plus fins peuvent être utilisés pour classer les implémentations existantes (voir Section 6.2). Nous avons démontré, par la création d'un médiateur (un négociateur tiers), une combinaison de modèles plus fins de négociation de contenu qui diffèrent des implémentations existantes (voir Section 7.4.1). Des parties de cette proposition ont été présentées à la 23e Conférence internationale sur l'ingénierie et la gestion des connaissances (EKAW 2022) [166].

Pour la Question de recherche 4, nous avons proposé une formalisation pour le problème de négociation de contenu distribué basée sur notre modèle CON-NEG. Nous avons proposé une approche basée sur l'utilisation des liens d'équivalence pour

découvrir des représentations potentiellement acceptables (voir Section 7.3). Nous avons proposé deux algorithmes qui utilisent notre approche pour deux dimensions de la négociation de contenu (le type de média et le profil dans la section 7.3.1 et la section 7.3.2 respectivement). Nous avons démontré l'applicabilité de notre approche dans la section 7.4.2, où nous nous sommes appuyés sur un service tiers pour découvrir des représentations potentiellement acceptables. Ensuite, dans la section 7.5, nous avons vérifié l'hypothèse selon laquelle l'utilisation d'une telle technique augmente le taux de découverte d'une représentation acceptable. Des parties de cette proposition ont été présentées au 1er atelier international sur la gestion des données pour les graphes de connaissances (DMKG 2023) à ESWC [169], ainsi qu'au 2ème atelier du collège doctoral pour la sémantique, le raisonnement et la coordination (SeReCo 2023) [165].

Pour la Question de recherche 5, nous avons proposé *CNCO*, une ontologie pour décrire les ressources négociables disponibles sur les serveurs web (voir Section 8.3). Nous avons introduit des vocabulaires de soutien pour *CNCO* pour les dimensions de négociation de contenu, les styles et les moyens de transmission des contraintes déjà connus (voir Section 8.3 et Annexe B). Les producteurs de ressources web négociables peuvent utiliser ces vocabulaires afin de créer des descriptions de ressources négociables. Nous avons démontré l'applicabilité de notre approche dans la section 8.4, où nous avons réutilisé des modèles et des technologies multi-agents existants. Ensuite, nous avons simulé un scénario dans lequel les agents logiciels utilisent des descriptions de ressources négociables pour savoir comment s'engager dans la négociation de contenu avec des serveurs web précédemment inconnus (voir Section 8.5). Des parties de cette proposition ont été présentées à la 23e Conférence internationale sur l'ingénierie et la gestion des connaissances (EKAW 2022) [166].

## C.6 Travaux futurs

Dans cette section, nous décrivons quelques-unes des perspectives que notre recherche ouvre pour de futures investigations :

Tout d'abord, dans nos travaux, nous nous sommes principalement concentrés sur les requêtes HTTP `GET`. Il convient toutefois de noter que, comme nous l'avons vu au Chapitre 3, la négociation de contenu peut également être effectuée dans le cadre de la méthode `POST`, par exemple, lorsque le client et le serveur négocient le contenu accepté à publier. Le modèle requête/réponse a également été utilisé pour illustrer nos contributions. Ainsi, une perspective est d'expérimenter le modèle de publier/s'abonner [68] et d'explorer les implications de l'utilisation d'une telle négociation de contenu plus fine et plus souple dans ces contextes.

Deuxièmement, dans notre étude des modèles plus fins de négociation de contenu, nous avons montré que différentes combinaisons ont été proposées et utilisées au fil des ans, dont l'une d'entre elles est la nôtre. L'une des perspectives de ce travail est d'effectuer une comparaison expérimentale de différents modèles plus fins en termes de performance, de complexité et de disponibilité des représentations.

Troisièmement, dans notre formalisation, nous nous sommes principalement concentrés sur le style proactif de négociation de contenu. Une perspective est de l'étendre à d'autres styles. Par exemple, dans le style réactif, où c'est plutôt le client qui prend la décision de sélection, la CN measure (voir Définition 4.6) ne devrait pas faire partie des données du serveur. Il en va de même si nous souhaitons formaliser les redirections HTTP.

Quatrièmement, nous n'avons pas étudié en profondeur le cas d'une négociation plus fine et plus souple de contenu adapté. Une piste, par exemple, consisterait à s'appuyer sur la réparation des violations de contraintes SHACL. Les recherches qui utilisent la programmation par ensembles de réponses [3] pourraient constituer un bon point de départ.

Cinquièmement, CNTF vise à mettre en évidence les solutions existantes pour les cas d'utilisation de la négociation de contenu, lorsqu'elles sont disponibles, ou à suggérer des moyens plausibles de répondre aux exigences communes (voir Annexe A). Nous prévoyons une mise en œuvre de chaque approche explorée et discutée dans cette thèse afin de faciliter leur adoption en disposant d'un espace expérimental. Un tel espace permettrait aux utilisateurs de tester différents styles de négociation de contenu, différentes dimensions et différentes solutions à certains cas d'utilisation sans avoir à les mettre en œuvre eux-mêmes.

Enfin, une autre perspective connexe consiste à proposer un service de validation de la négociation de contenu similaire à Vapour [24] et au validateur Memento [116], mais avec des dimensions supplémentaires de négociation de contenu. Le service créerait en outre une description de la ressource à négocier que la personne validant la ressource pourrait publier si elle le souhaite. Et si le propriétaire donne son autorisation, cette description peut être ajoutée à un KG qui, à terme, peut être utilisé par un moteur de recherche global de ressources négociables similaire à celui présenté au Chapitre 8.

# Bibliography


[1] Abhayaratna, J., Atkinson, R.: Negotiating Content by Coordinate Reference System (CRS) (Jul 2 2019), https://github.com/opengeospatial/conneg-by-crs

[2] Ahmetaj, S., David, R., Ortiz, M., Polleres, A., Shehu, B., Šimkus, M.: Reasoning about Explanations for Non-validation in SHACL. In: Proceedings of the 18th International Conference on Principles of Knowledge Representation and Reasoning. pp. 12–21 (Nov 2021), https://doi.org/10.24963/kr.2021/2

[3] Ahmetaj, S., David, R., Polleres, A., Simkus, M.: Repairing SHACL Constraint Violations Using Answer Set Programming. In: The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13489, pp. 375–391. Springer (Oct 2022), https://doi.org/10.1007/978-3-031-19433-7_22

[4] Akhtar, W., Kopecký, J., Krennwallner, T., Polleres, A.: XSPARQL: traveling between the XML and RDF worlds - and avoiding the XSLT pilgrimage. In: The Semantic Web: Research and Applications, 5th European Semantic Web Conference, ESWC 2008, Tenerife, Canary Islands, Spain, June 1-5, 2008, Proceedings. Lecture Notes in Computer Science, vol. 5021, pp. 432–447. Springer (2008), https://doi.org/10.1007/978-3-540-68234-9_33

[5] Andrews, G.R.: Paradigms for Process Interaction in Distributed Programs. ACM Comput. Surv. **23**(1), 49–90 (Mar 1991), https://doi.org/10.1145/103162.103164

[6] Ankolekar, A., Vrandecic, D.: Personalizing web surfing with semantically enriched peronal profiles. In: Proceedings of the Semantic Web Personalization Workshop (Jun 2006), https://www.aifb.kit.edu/images/2/26/2006_1201_Ankolekar_Personalizing_w_1.pdf

[7] Atkinson, R.: OGC Definitions Server (2020), https://www.ogc.org/resources/def-server/

[8] Atkinson, R., Car, N.J.: The Profiles Vocabulary. W3C Working Group Note, W3C (Dec 18 2019), https://www.w3.org/TR/2019/NOTE-dx-prof-20191218/

[9] Atkinson, R., Car, N.J.: The Profile Wiz Tool (2021), https://github.com/RDFLib/profilewiz

[10] Attouche, L., Baazizi, M.A., Colazzo, D., Ghelli, G., Sartiani, C., Scherzinger, S.: Validation of Modern JSON Schema: Formalization and Complexity. CoRR **abs/2307.10034** (2023), https://doi.org/10.48550/arXiv.2307.10034

[11] Bachmann-Gmür, R.: HTTP Extension For Vocabulary Negotiation (Aug 1 2006), https://www.w3.org/wiki/VocabularyNegotiation

[12] Beek, W., Raad, J., Wielemaker, J., van Harmelen, F.: sameAs.cc: The Closure of 500M owl: sameAs Statements. In: The Semantic Web - 15th International Conference, ESWC 2018, Heraklion, Crete, Greece, June 3-7, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10843, pp. 65–80. Springer (2018), https://doi.org/10.1007/978-3-319-93417-4_5

[13] Berners-Lee, T.: Information Management: A Proposal. Tech. rep., CERN (Mar 1989), https://www.w3.org/History/1989/proposal.html

[14] Berners-Lee, T.: WorldWideWeb - Executive Summary (Aug 6 1991), https://www.w3.org/People/Berners-Lee/1991/08/art-6487.txt

[15] Berners-Lee, T.: Basic HTTP as Defined in 1992 (1992), https://www.w3.org/Protocols/HTTP/HTTP2.html

[16] Berners-Lee, T.: HTTP Negotiation Algoritm (1992), https://www.w3.org/Protocols/HTTP/Negotiation.html

[17] Berners-Lee, T., Cailliau, R., Luotonen, A., Nielsen, H.F., Secret, A.: The World-Wide Web. Commun. ACM **37**(8), 76–82 (1994), https://doi.org/10.1145/179606.179671

[18] Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifiers (URI): Generic Syntax. RFC 2396, Network Working Group (Aug 1998), https://doi.org/10.17487/RFC2396

[19] Berners-Lee, T., Fielding, R.T., Nielsen, H.F.: Hypertext Transfer Protocol - HTTP/1.0. RFC 1945, Network Working Group (1996), https://doi.org/10.17487/RFC1945

[20] Berners-Lee, T., Fielding, R.T., Masinter, L.: Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Internet Engineering Task Force (Jan 2005), http://tools.ietf.org/html/rfc3986

[21] Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. Scientific American **284**(5), 34–43 (May 2001), https://www.scientificamerican.com/article/the-semantic-web/

[22] Berners-Lee, T., Masinter, L., McCahill, M.: Uniform Resource Locators (URL). RFC 1738, Network Working Group (Dec 1994), https://doi.org/10.17487/RFC1738

[23] Berrueta, D., Phipps, J., Miles, A., Baker, T., Swick, R.: Best Practice Recipes for Publishing RDF Vocabularies. W3C Working Group Note, W3C (Aug 28 2008), https://www.w3.org/TR/swbp-vocab-pub/

[24] Berrueta Muñoz, D., Fernández, S., Frade, I.: Cooking HTTP Content Negotiation with Vapour. CEUR Workshop Proceedings (Jan 2008), https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=85f94c8ece062dad9f96783d42b4e105791038cf

[25] Bianchi, T.: Distribution of Bot and Human Web Traffic Worldwide From 2014 To 2022 (Jul 17 2023), https://www.statista.com/statistics/1264226/human-and-bot-web-traffic-share/

[26] Birkholz, H., Vigano, C., Bormann, C.: Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures. RFC 8610, IETF (Jun 2019), https://doi.org/10.17487/RFC8610

[27] Boissier, O., Bordini, R.H., Hübner, J.F., Ricci, A., Santi, A.: Multi-agent oriented programming with jacamo. Sci. Comput. Program. **78**(6), 747–761 (Jun 1 2013), https://doi.org/10.1016/j.scico.2011.10.004

[28] Bormann, C., Hoffman, P.E.: Concise Binary Object Representation (CBOR). RFC 8949, IETF (2020), https://doi.org/10.17487/RFC8949

[29] Braden, R.T.: Requirements for internet hosts - application and support. RFC 1123, IETF (Oct 1989), https://doi.org/10.17487/RFC1123

[30] Bray, T.: The JavaScript Object Notation (JSON) Data Interchange Format. RFC 8259, IETF (Dec 2017), https://doi.org/10.17487/RFC8259

[31] Bray, T., Paoli, J., Sperberg-McQueen, C.M.: Extensible Markup Language (XML) 1.0. W3C Recommendation, W3C (Feb 10 1998), http://www.w3.org/TR/WD-xml-970807

[32] Brickley, D., Guha, R.V.: RDF Schema 1.1. W3C Recommendation, W3C (Feb 25 2014), http://www.w3.org/TR/2014/REC-rdf-schema-20140225/

[33] Brickley, D., Miller, L.: FOAF Vocabulary Specification (May 1 2004), http://xmlns.com/foaf/spec/

[34] van den Brink, L., Barnaghi, P.M., Tandy, J., Atemezing, G., Atkinson, R., Cochrane, B., Fathy, Y., García-Castro, R., Haller, A., Harth, A., Janowicz, K., Kolozali, S., van Leeuwen, B., Lefrançois, M., Lieberman, J., Perego, A., Le Phuoc, D., Roberts, B., Taylor, K., Troncy, R.: Best Practices for Publishing, Retrieving, and Using Spatial Data on the Web. Semantic Web Journal **10**(1), 95–114 (2019), https://doi.org/10.3233/SW-180305

[35] Brown, A., Fuchs, M., Robie, J., Wadler, P.: XML Schema: Formal Description. W3c working draft, W3C (Sep 25 2001), https://www.w3.org/TR/2001/WD-xmlschema-formal-20010925/

[36] Butler, M.H.: Implementing Content Negotiation Using CC/PP and WAP UAProf. Tech. Rep. HPL-2001-190, HP Lab (2001), https://www.hpl.hp.com/techreports/2001/HPL-2001-190.html

[37] Calzarossa, M.C., Massari, L.: Analysis of Header Usage Patterns of HTTP Request Messages. In: 2014 IEEE Intl Conf on High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC,CSS,ICESS). pp. 847–853 (2014), https://ieeexplore.ieee.org/abstract/document/7056844

[38] Car, N.J.: PHP ConnegP (2019), https://github.com/nicholascar/php-connegp/, (Version 0.7)

[39] Car, N.J.: Media Types Register (2021), https://github.com/nicholascar/mediatypes-service

[40] Car, N.J.: The Python Linked Data API (pyLDAPI) (2021), https://github.com/rdflib/pyLDAPI, (Version 4.2)

[41] Car, N.J.: CKAN DCAT Plugin (2022), https://github.com/ckan/ckanext-dcat, (Version 1.4.0)

[42] Choudhury, N.: World Wide Web and Its Journey from Web 1.0 to Web 4.0. International Journal of Computer Science and Information Technologies **5**(6), 8096–8100 (2014), https://ijcsit.com/docs/Volume%205/vol5issue06/ijcsit20140506265.pdf

[43] Ciortea, A., Mayer, S., Gandon, F., Boissier, O., Ricci, A., Zimmermann, A.: A Decade in Hindsight: The Missing Bridge Between Multi-Agent Systems and the World Wide Web. In: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019. pp. 1659–1663. International Foundation for Autonomous Agents and Multiagent Systems (2019), http://dl.acm.org/citation.cfm?id=3331893

[44] Coates, T., Connolly, D., Dack, D., Daigle, L.L., Denenberg, R., Dürst, M.J., Grosso, P., Hawke, S., Iannella, R., Klyne, G., et al.: URIs, URLs, and URNs: Clarifications and Recommendations 1.0. W3C Note, W3C (Sep 21 2001), https://www.w3.org/TR/2001/NOTE-uri-clarification-20010921/

[45] Corby, O., Faron-Zucker, C.: STTL - A sparql-based transformation language for RDF. In: WEBIST 2015 - Proceedings of the 11th International Conference on Web Information Systems and Technologies, Lisbon, Portugal, 20-22 May, 2015. pp. 466–476. SciTePress (2015), https://doi.org/10.5220/0005450604660476

[46] Corman, J., Florenzano, F., Reutter, J.L., Savkovic, O.: SHACL2SPARQL: Validating a SPARQL Endpoint Against Recursive SHACL Constraints. In: Proceedings of the ISWC 2019 Satellite Tracks (Posters & Demonstrations, Industry, and Outrageous Ideas), Auckland, New Zealand, October 26-30, 2019. CEUR Workshop Proceedings, vol. 2456, pp. 165–168. CEUR-WS.org (Oct 2019), https://ceur-ws.org/Vol-2456/paper43.pdf

[47] Corman, J., Reutter, J.L., Savković, O.: Semantics and Validation of Recursive SHACL. In: The Semantic Web - ISWC 2018 - 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I. Lecture Notes in Computer Science, vol. 11136, pp. 318–336. Springer (2018), https://doi.org/10.1007/978-3-030-00671-6_19

[48] Cyganiak, R., Hyland-Wood, D., Lanthaler, M.: RDF 1.1 Concepts and Abstract Syntax. W3C Recommendation, W3C (Feb 25 2014), https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/

[49] Dimou, A., Sande, M.V., Colpaert, P., Verborgh, R., Mannens, E., de Walle, R.V.: RML: A generic language for integrated RDF mappings of heterogeneous data. In: Proceedings of the Workshop on Linked Data on the Web co-located with the 23rd International World Wide Web Conference (WWW 2014), Seoul, Korea, April 8, 2014. CEUR Workshop Proceedings, vol. 1184. CEUR-WS.org (2014), https://ceur-ws.org/Vol-1184/ldow2014_paper_01.pdf

[50] Ding, L., Shinavier, J., Shangguan, Z., McGuinness, D.L.: SameAs Networks and Beyond: Analyzing Deployment Status and Implications of owl:sameAs in Linked Data. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. pp. 145–160 (2010), https://doi.org/10.1007/978-3-642-17746-0_10

[51] Doglio, F.: Working with Modules. Apress (2018), https://doi.org/10.1007/978-1-4842-3715-1_5

[52] Dürst, M.J., Suignard, M.: Internationalized Resource Identifiers (IRIs). RFC 3987, Network Working Group (Jan 2005), https://doi.org/10.17487/RFC3987

[53] Dusseault, L., Snell, J.M.: PATCH Method for HTTP. RFC 5789, IETF (Mar 2010), https://doi.org/10.17487/RFC5789

[54] European Telecommunications Standards Institute: ETSI TS 101 438 V7.2.0. Tech. rep., European Telecommunications Standards Institute (Apr 2000), https://openconnectivity.org/specs/OCF_Core_Specification_v2.2.5.pdf

[55] Farias Lóscio, B., Burle, C., Calegari, N.: Data on the Web Best Practices. W3C Recommendation, W3C (Jan 31 2017), https://www.w3.org/TR/2017/REC-dwbp-20170131/

[56] Feigenbaum, L., Williams, G.T., Clark, K.G., Torres, E.: SPARQL 1.1 Protocol. W3C Recommendation, W3C (Mar 21 2013), https://www.w3.org/TR/2013/REC-sparql11-protocol-20130321/

[57] Felin, R., Faron, C., Tettamanzi, A.G.B.: A Framework to Include and Exploit Probabilistic Information in SHACL Validation Reports. In: The Semantic Web - 20th International Conference, ESWC 2023, Hersonissos, Crete, Greece, May 28 - June 1, 2023, Proceedings. Lecture Notes in Computer Science, vol. 13870, pp. 91–104. Springer (2023), https://doi.org/10.1007/978-3-031-33455-9_6

[58] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., Masinter, L., Leach, P., Berners-Lee, T.: Hypertext Transfer Protocol - HTTP/1.1. RFC 2616, IETF (Jun 1999), https://doi.org/10.17487/RFC2616

[59] Fielding, R., Reschke, J.: Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. RFC 7231, IETF (Jun 2014), https://doi.org/10.17487/RFC7231

[60] Fielding, R.T., Gettys, J., Mogul, J.C., Nielsen, H.F., Berners-Lee, T.: Hypertext Transfer Protocol - HTTP/1.1. RFC 2068, Network Working Group (Jan 1997), https://doi.org/10.17487/RFC2068

[61] Fielding, R.T., Nottingham, M., Reschke, J.F.: HTTP Semantics. RFC 9110, IETF (Jun 2022), https://doi.org/10.17487/RFC9110

[62] Fielding, R.T.: Architectural Styles and the Design of Network-Based Software Architectures. Ph.D. thesis, University of California, Irvine (2000), https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[63] Freed, N., Borenstein, N.S.: Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. RFC 2046, Network Working Group (Nov 1996), https://doi.org/10.17487/RFC2046

[64] Freed, N., Klensin, J.C., Hansen, T.: Media Type Specifications and Registration Procedures. RFC 6838, IETF (Jan 2013), https://doi.org/10.17487/RFC6838

[65] Gandon, F.: Finding the Critical MAS(S): Resources and Representations Needed for Weaving a Web that Hosts Linked Multi-Agent Systems. In: Dagstuhl Seminar 23081 “Agents on the Web” (2023), https://inria.hal.science/hal-03989703/document

[66] Gao, S.S., Sperberg-McQueen, C.M., Thompson, H.: W3C XML schema definition language (XSD) 1.1 part 1: Structures. W3C Recommendation, W3C (Apr 5 2012), https://www.w3.org/TR/xmlschema11-1/

[67] Gayo, J.E.L., Prud'hommeaux, E., Boneva, I., Kontokostas, D.: Validating RDF Data. Synthesis Lectures on the Semantic Web: Theory and Technology, Morgan & Claypool Publishers (2017), https://doi.org/10.2200/S00786ED1V01Y201707WBE016

[68] Genestoux, J., Parecki, A.: WebSub. W3C Recommendation, W3C (Jan 23 2018), https://www.w3.org/TR/websub/

[69] Glaser, H.: sameas.org , Interlinking the Web of Data (2009), http://sameas.org/

[70] Gleim, L.C., Decker, S.: Timestamped URLs as persistent identifiers. In: MEPDaW@ ISWC. pp. 11–16 (2020), https://mepdaw-ws.github.io/2020/accepted-papers/MEPDaW_2020_paper_8.pdf

[71] Glimm, B., Ogbuji, C.: SPARQL 1.1 Entailment Regimes. W3C Recommendation, W3C (Mar 21 2013), http://www.w3.org/TR/2013/REC-sparql11-entailment-20130321/

[72] Goto, S.: Vocabulary Negotiation (Sep 1 2015), https://code.sgo.to/2015/09/01/vocabulary-negotiation.html

[73] Group, W.O.W.: OWL 2 Web Ontology Language Document Overview (Second Edition). W3C Recommendation, W3C (Dec 11 2012), https://www.w3.org/TR/owl-overview/

[74] Gruber, T.R.: A translation approach to portable ontology specifications. Knowledge acquisition **5**(2), 199–220 (1993), https://www.sciencedirect.com/science/article/pii/S1042814383710083

[75] Guenther, R.S.: Mods: the metadata object description schema. Portal: libraries and the academy **3**(1), 137–150 (2003), https://www.loc.gov/standards/mods/3.1guenther.pdf

[76] Gupta, L.: Content Negotiation in REST (Sep 30 2021), https://restfulapi.net/content-negotiation/

[77] Gupta, L.: How to Version a REST API? (Dec 30 2022), https://restfulapi.net/versioning/

[78] Haase, P., Broekstra, J., Eberhart, A., Volz, R.: A Comparison of RDF Query Languages. In: The Semantic Web - ISWC 2004: Third International Semantic Web Conference,Hiroshima, Japan, November 7-11, 2004. Proceedings. Lecture Notes in Computer Science, vol. 3298, pp. 502–517. Springer (2004), https://doi.org/10.1007/978-3-540-30475-3_35

[79] Halpin, H., Hayes, P.J., McCusker, J.P., McGuinness, D.L., Thompson, H.S.: When owl:sameAs Isn't the Same: An Analysis of Identity in Linked Data. In: The Semantic Web - ISWC 2010 - 9th International Semantic Web Conference, ISWC 2010, Shanghai, China, November 7-11, 2010, Revised Selected Papers, Part I. pp. 305–320 (Nov 2010), https://doi.org/10.1007/978-3-642-17746-0_20

[80] Hammond, T., Pasin, M.: Nature.com Core Ontology (npg) V1.0.41 (Apr 30 2015), https://lov.linkeddata.es/dataset/lov/vocabs/npg

[81] Hardie, T.: Scenarios for the Delivery of Negotiated Content. Internet-Draft draft-ietf-http-negotiate-scenario-02, IETF (Nov 1997), https://datatracker.ietf.org/doc/draft-ietf-http-negotiate-scenario/02/, Work in Progress

[82] Hayes, P., Patel-Schneider, P.: RDF 1.1 Semantics. W3C Recommendation, W3C (Feb 25 2014), https://www.w3.org/TR/2014/REC-rdf11-mt-20140225/

[83] Holtman, K., Mutz, A.H.: HTTP Remote Variant Selection Algorithm - RVSA/1.0. RFC 2296, Network Working Group (Mar 1998), https://doi.org/10.17487/RFC2296

[84] Holtman, K., Mutz, A.H.: Transparent Content Negotiation in HTTP. RFC 2295, Network Working Group (Mar 1998), https://doi.org/10.17487/RFC2295

[85] Holtman, K., Mutz, A.H., Hardie, T.: Media Feature Tag Registration Procedure. RFC 2506, Network Working Group (Mar 1999), https://doi.org/10.17487/RFC2506

[86] Horrocks, I., Patel-Schneider, P.F., van Harmelen, F.: From SHIQ and RDF to OWL: the making of a Web Ontology Language. Journal of Web Semantics **1**(1), 7–26 (2003), https://www.sciencedirect.com/science/article/pii/S1570826803000027

[87] Huber, R., D'Onofrio, C., Devaraju, A., Klump, J., Loescher, H.W., Kindermann, S., Guru, S., Grant, M., Morris, B., Wyborn, L., et al.: Integrating data and analysis technologies within leading environmental research infrastructures: Challenges and approaches. Ecological Informatics **61**, 101245 (2021), https://www.sciencedirect.com/science/article/pii/S1574954121000364

[88] Jacobs, I., Walsh, N.: Architecture of the World Wide Web, Volume One. W3C Recommendation, W3C (Dec 15 2004), http://www.w3.org/TR/2004/REC-webarch-20041215/

[89] Jaffri, A.: Searching on the Open Semantic Web using a URI Identity Management Approach (2007), https://eprints.soton.ac.uk/264758/1/poster-3.pdf

[90] Juviler, J.: API Versioning: A Marketer's Guide (Jun 23 2022), https://blog.hubspot.com/website/api-versioning

[91] Kelly, M., Alam, S., Nelson, M.L., Weigle, M.C.: Client-Assisted Memento Aggregation Using the Prefer Header. In: Proceedings of the 2018 Web Archiving & Digital Libraries Workshop (WADL 2018), June 6, 2018, Fort Worth, Texas, USA. pp. 5–6. Virginia Tech University Libraries (Jun 2018), https://vtechworks.lib.vt.edu/bitstream/handle/10919/97988/WADL2018.pdf

[92] Klyne, G.: Protocol-independent Content Negotiation Framework. RFC 2703, Network Working Group (Sep 1999), https://doi.org/10.17487/RFC2703

[93] Klyne, G., Iwazaki, R., Crocker, D.: Content Negotiation for Messaging Services based on Email. RFC 3297, Network Working Group (Jul 2002), https://doi.org/10.17487/RFC3297

[94] Klyne, G., Reynolds, F., Woodrow, C., Ohto, H., Hjelm, J., Butler, M.H., Tran, L.: Composite Capability/Preference Profiles (CC/PP): Structure and Vocabularies 1.0. W3C Recommendation, W3C (Jan 15 2004), https://www.w3.org/TR/2004/REC-CCPP-struct-vocab-20040115/

[95] Knublauch, H.: DASH Data Shapes Vocabulary. Unofficial Draft, TopQuadrant (Sep 30 2021), https://datashapes.org/dash

[96] Knublauch, H., Hendler, J.A., Idehen, K.: SPIN – Overview and Motivation. W3c member submission, W3C (Feb 22 2011), https://www.w3.org/submissions/2011/SUBM-spin-overview-20110222/

[97] Knublauch, H., Kontokostas, D.: Shapes Constraint Language (SHACL). W3C Recommendation, W3C (Jul 20 2017), https://www.w3.org/TR/2017/REC-shacl-20170720/

[98] Koch, J., Velasco, C.A., Ackermann, P.: HTTP vocabulary in RDF 1.0. W3c working draft, W3C (Feb 2 2017), https://www.w3.org/TR/2017/NOTE-HTTP-in-RDF10-20170202/

[99] Kunze, J.A., Bermès, E.: The ARK Identifier Scheme. Internet-Draft draft-kunze-ark-36, Internet Engineering Task Force (Apr 25 2023), https://datatracker.ietf.org/doc/draft-kunze-ark/37/, Work in Progress

[100] Kunze, J.A., Rodgers, R.P.C.: The ARK Identifier Scheme. Internet-Draft draft-kunze-ark-09, Internet Engineering Task Force (Feb 19 2005), https://datatracker.ietf.org/doc/html/draft-kunze-ark-09, Work in Progress

[101] Lagoze, C., Van de Sompel, H., Nelson, M., Warner, S.: The Open Archives Initiative Protocol for Metadata Harvesting. Tech. rep., Open Archives Initiative (Jun 14 2002), http://www.openarchives.org/OAI/openarchivesprotocol.html, (Version 2.0)

[102] Lanthaler, M.: The Profile URI Registry. RFC **7284**, 1–5 (2014), https://doi.org/10.17487/RFC7284

[103] Lanthaler, M.: Hydra Core Vocabulary. W3C Unofficial Draft, W3C (Jul 13 2021), https://www.hydra-cg.com/spec/latest/core/

[104] Lefrançois, M.: RDF Presentation and Correct Content Conveyance for Legacy Services and the Web of Things. In: Proceedings of the 8th International Conference on the Internet of Things, IOT 2018, Santa Barbara, CA, USA, October 15-18, 2018. pp. 43:1–43:8. ACM Press (Oct 2018), https://doi.org/10.1145/3277593.3277618

[105] Lefrançois, M.: Interopérabilité sémantique libérale pour les services et les objets. In: 17ème Journées Francophones Extraction et Gestion des Connaissances, EGC 2017, 24-27 Janvier 2017, Grenoble, France. RNTI, vol. E-33, pp. 81–92. Éditions RNTI (2017), http://editions-rnti.fr/?inprocid=1002271

[106] Lefrançois, M., Zimmermann, A., Bakerally, N.: Flexible RDF generation from RDF and heterogeneous data sources with sparql-generate. In: Knowledge Engineering and Knowledge Management - EKAW 2016 Satellite Events, EKM and Drift-an-LOD, Bologna, Italy, November 19-23, 2016, Revised Selected Papers. Lecture Notes in Computer Science, vol. 10180, pp. 131–135. Springer (2016), https://doi.org/10.1007/978-3-319-58694-6_16

[107] Lefrançois, M.: The RDF Presentation Ontology (rdfp) (Oct 24 2016), https://lov.linkeddata.es/dataset/lov/vocabs/rdfp

[108] Leinberger, M., Seifer, P., Rienstra, T., Lämmel, R., Staab, S.: Deciding SHACL Shape Containment Through Description Logics Reasoning. In: The Semantic Web - ISWC 2020 - 19th International Semantic Web Conference, Athens, Greece, November 2-6, 2020, Proceedings, Part I. Lecture Notes in Computer Science, vol. 12506, pp. 366–383. Springer (Nov 2020), https://doi.org/10.1007/978-3-030-62419-4_21

[109] Lemlouma, T., Layaïda, N.: NAC: A Basic Core for the Adaptation and Negotiation of Multimedia Services. Opera Project, INRIA (2001), https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=3a48e0bd1d62b59f53231090d7808494a421c572

[110] Lemlouma, T., Layaïda, N.: Universal Profiling for Content Negotiation and Adaptation in Heterogeneous Environments. In: W3C Workshop on Delivery Context. pp. 4–5 (2002), https://www.w3.org/2002/02/DIWS/submission/tlemlouma-W3CPositionPaper03-2002.htm

[111] Lerouge, S.: Personalizing Quality Aspects for Video Communication in Constrained Heterogeneous Environments. Ph.D. thesis, Ghent University (Nov 24 2006), https://backoffice.biblio.ugent.be/download/467760/1878996

[112] Lirzin, M., Markhoff, B.: Towards an Ontology of HTTP Interactions. CoRR **abs/2007.13475** (2020), https://arxiv.org/abs/2007.13475

[113] Lohmann, S., Link, V., Marbach, E., Negru, S.: WebVOWL: Web-based Visualization of Ontologies. In: Knowledge Engineering and Knowledge Management - EKAW 2014 Satellite Events, VISUAL, EKM1, and ARCOE-Logic, Linköping, Sweden, November 24-28, 2014. Revised Selected Papers. Lecture Notes in Computer Science, vol. 8982, pp. 154–158. Springer (2014), https://doi.org/10.1007/978-3-319-17966-7_21

[114] Lum, W.Y., Lau, F.C.M.: On Balancing Between Transcoding Overhead and Spatial Consumption in Content Adaptation. In: Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking, MOBICOM 2002, Atlanta, Georgia, USA, September 23-28, 2002. pp. 239–250. ACM (Sep 2002), https://doi.org/10.1145/570645.570675

[115] Lum, W.Y., Lau, F.C.M.: User-Centric Content Negotiation for Effective Adaptation Service in Mobile Computing. IEEE Transaction on Software Engineering **29**(12), 1100–1111 (Dec 2003), https://doi.org/10.1109/TSE.2003.1265524

[116] Mahanama, B., Balakireva, L., Jayarathna, S., Nelson, M.L., Klein, M.: Memento Validator: A Toolset for Memento Compliance Testing. In: JCDL '22: The ACM/IEEE Joint Conference on Digital Libraries in 2022, Cologne, Germany, June 20 - 24, 2022. p. 43. ACM (2022), https://doi.org/10.1145/3529372.3533297

[117] Manola, F., Miller, E.: RDF Primer. W3C Recommendation, W3C (Feb 10 2004), http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

[118] MarkLogic Corporation: MarkLogic Server - REST Application Developer's Guide. Tech. rep., MarkLogic Corporation (May 2019), https://docs.marklogic.com/guide/rest-dev.pdf

[119] Masinter, L., Wing, D., Mutz, A.H., Holtman, K.: Media Features for Display, Print, and Fax. RFC 2534, Network Working Group (Mar 1999), https://doi.org/10.17487/RFC2534

[120] McGuinness, D.L., van Harmelen, F.: OWL Web Ontology Language, Overview. W3C Recommendation, W3C (Feb 10 2004), http://www.w3.org/TR/2004/REC-owl-features-20040210/

[121] Mikael K., B., Frédéric, G.: Upper Mapping and Binding Exchange Layer (umbel) (Aug 28 2014), https://lov.linkeddata.es/dataset/lov/vocabs/umbel

[122] Miles, A., Bechhofer, S.: SKOS Simple Knowledge Organization System, Reference. W3C Recommendation, W3C (Aug 18 2009), http://www.w3.org/TR/2009/REC-skos-reference-20090818/

[123] Motik, B., Cuenca-Grau, B., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C.: OWL 2 Web Ontology Language Profiles (Second Edition). W3C Recommendation, W3C (Dec 11 2012), http://www.w3.org/TR/2012/REC-owl2-profiles-20121211/

[124] Mountantonakis, M., Tzitzikas, Y.: LODsyndesis: The Biggest Knowledge Graph of the Linked Open Data Cloud that Includes all Inferred Equivalence Relationships. ERCIM News **2018**(114) (2018), https://users.ics.forth.gr/~mountant/publications/LODsyndesis_ERCIM114.pdf

[125] Murata, M., Laurent, S.S., Kohn, D.: XML Media Types. RFC 3023, Network Working Group (Jan 2001), https://doi.org/10.17487/RFC3023

[126] Naylor, D., Finamore, A., Leontiadis, I., Grunenberger, Y., Mellia, M., Munafò, M.M., Papagiannaki, K., Steenkiste, P.: The Cost of the "S" in HTTPS. In: Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, CoNEXT 2014, Sydney, Australia, December 2-5, 2014. pp. 133–140. ACM (2014), https://doi.org/10.1145/2674005.2674991

[127] Nebert, D., Voges, U., Vretanos, P., Bigagli, L., Westcott, B.: OGC® Catalogue Services 3.0 Specification - HTTP Protocol Binding. Tech. rep., Open Geospatial Consortium (Jun 10 2016), https://docs.ogc.org/is/12-176r7/12-176r7.html

[128] Nelson, T.H.: Complex Information Processing: A File Structure for the Complex, the Changing and the Indeterminate. In: Proceedings of the 20th national conference, ACM 1965, Cleveland, Ohio, USA, 1965. pp. 84–100. ACM (1965), https://doi.org/10.1145/800197.806036

[129] Nottingham, M.: Web Linking. RFC 8288, IETF (Oct 2017), https://doi.org/10.17487/RFC8288

[130] Nottingham, M., Wilde, E., Dalal, S.: Problem details for HTTP apis. RFC, IETF (Jul 2023), https://doi.org/10.17487/RFC9457

[131] Ohto, H., Hjelm, J.: CC/PP Exchange Protocol Based on HTTP Extension Framework. W3C Note, W3C (Jun 24 1999), https://www.w3.org/TR/NOTE-CCPPexchange

[132] Open Mobile Alliance: User Agent Profile. Tech. rep., Open Mobile Alliance (Feb 06 2006), https://www.openmobilealliance.org/release/UAProf/V2_0_1-20070625-A/OMA-TS-UAProf-V2_0-20060206-A.pdf, approved Version 2.0

[133] Pattan, B.J., Raghavendran, R.: System and Method for Transmitting System Messages Insession Initiation Protocol (May 26 2015), https://patents.google.com/patent/US9043394/en, uS Patent 9,043,394

[134] Phillips, A., Davis, M.: Matching of Language Tags. RFC 4647, Network Working Group (Sep 2006), https://doi.org/10.17487/RFC4647

[135] Postel, J.: Simple Mail Transfer Protocol. RFC 788, Network Working Group (1981), https://doi.org/10.17487/RFC0788

[136] Postel, J., Reynolds, J.K.: File Transfer Protocol. RFC 959, Network Working Group (1985), https://doi.org/10.17487/RFC0959

[137] Prud'hommeaux, E., Boneva, I., Gayo, J.E.L., Kellogg, G.: Shape Expressions Language 2.1. W3C Community Group Report, W3C (2019), http://shex.io/shex-semantics-20191008/

[138] Prud'hommeaux, E., Carothers, G.: RDF 1.1 Turtle - Terse RDF Triple Language. W3C Recommendation, W3C (Feb 25 2014), http://www.w3.org/TR/2014/REC-turtle-20140225/

[139] Prud'hommeaux, E., Gayo, J.E.L., Solbrig, H.R.: Shape expressions: an RDF validation and transformation language. In: Proceedings of the 10th International Conference on Semantic Systems, SEMANTiCS 2014, Leipzig, Germany, September 4-5, 2014. pp. 32–40. ACM (2014), https://doi.org/10.1145/2660517.2660523

[140] Pullmann, J., Atkinson, R., Isaac, A., Faniel, I.: Dataset Exchange Use Cases and Requirements. W3C Working Group Note, W3C (Jan 17 2019), https://www.w3.org/TR/2019/NOTE-dcat-ucr-20190117

[141] Raad, J., Pernelle, N., Saïs, F., Beek, W., van Harmelen, F.: The sameAs Problem: A Survey on Identity Management in the Web of Data. CoRR **abs/1907.10528** (2019), http://arxiv.org/abs/1907.10528

[142] Research, R.: What's Wrong with REST (Apr 5 2021), https://www.riomhaire.com/post/rest-schemas/

[143] Reynolds, D.: The Organization Ontology. W3C Recommendation, W3C (Jan 16 2014), https://www.w3.org/TR/2014/REC-vocab-org-20140116/

[144] Ridley, B.: Open Source PHP Content Negotiation Library (2013), http://ptlis.net/source/php/content-negotiation/

[145] Rohde, P.D., Iglesias, E., Vidal, M.E.: SHACL-ACL: Access Control with SHACL. In: ESWC 2023 Posters, Demos (2023), https://2023.eswc-conferences.org/wp-content/uploads/2023/05/paper_Rohde_2023_SHACL-ACL.pdf

[146] Romary, L., Lundberg, S.: The 'application/tei+xml' Media Type. RFC 6129, IETF (Feb 2011), https://doi.org/10.17487/RFC6129

[147] Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach (4th Edition). Pearson (2020), http://aima.cs.berkeley.edu/

[148] Ryman, A.G., Le Hors, A., Speicher, S.: OSLC Resource Shape: A Language for Defining Constraints on Linked Data. LDOW **996** (2013), https://ceur-ws.org/Vol-996/papers/ldow2013-paper-02.pdf

[149] Saint-Andre, P., Klensin, J.C.: Uniform Resource Names (URNs). RFC 8141, IETF (2017), https://doi.org/10.17487/RFC8141

[150] Sauermann, L., Cyganiak, R.: Cool URIs for the Semantic Web. W3C Note, W3C (Dec 3 2008), https://www.w3.org/TR/cooluris/

[151] Schadow, G., McDonald, C.J.: The Unified Code for Units of Measure. Tech. rep., Regenstrief Institute (Nov 21 2017), https://ucum.org/ucum, version: 2.1

[152] Schmalenbach, K.: RDF Browser (Oct 5 2022), https://addons.mozilla.org/en-US/firefox/addon/rdf-browser/

[153] Schreiber, G., Raimond, Y.: RDF 1.1 Primer. Tech. rep., W3C (Jun 24 2014), https://www.w3.org/TR/2014/NOTE-rdf11-primer-20140624/#section-graph-syntax

[154] Seaborne, A., Prud'hommeaux, E.: SPARQL Query Language for RDF. W3C Recommendation, W3C (Jan 15 2008), http://www.w3.org/TR/rdf-sparql-query/

[155] Seshan, S., Stemm, M., Katz, R.H.: Benefits of Transparent Content Negotiation in HTTP. In: Proceedings of the IEEE Globcom 98 Internet Mini-Conference. Citeseer (1998), https://www.researchgate.net/publication/228721179_Benefits_of_transparent_content_negotiation_in_HTTP

[156] Shelby, Z., Hartke, K., Bormann, C.: The Constrained Application Protocol (CoAP). Tech. rep., Internet Engineering Task Force (Jun 2014), http://tools.ietf.org/html/rfc7252

[157] Snell, J.: Prefer Header for HTTP. RFC 7240, IETF (Jun 2014), https://www.rfc-editor.org/rfc/rfc7240

[158] Solanki, M., Mader, C.: Towards Maintainable Constraint Validation and Repair for Taxonomies: The PoolParty Approach. In: Proceedings of the 7th International Workshop on Consuming Linked Data co-located with 15th International Semantic Web Conference, COLD@ISWC 2015, Kobe, Japan, October 18, 2016. CEUR Workshop Proceedings, vol. 1666. CEUR-WS.org (Oct 2016), http://ceur-ws.org/Vol-1666/paper-06.pdf

[159] Van de Sompel, H., Nelson, M.L., Sanderson, R.: HTTP Framework for Time-Based Access to Resource States - Memento. RFC 7089, IETF (2013), https://www.rfc-editor.org/rfc/rfc7089.txt

[160] Svensson, L.G.: An http Header for Metadata Schema Negotiation. In: W3C Workshop on Smart Descriptions & Smarter Vocabularies (SDSVoc). W3C (Nov 2016), https://www.w3.org/2016/11/sdsvoc/SDSVoc16_paper_14

[161] Svensson, L.G., Atkinson, R., Car, N.J.: Content Negotiation by Profile. W3C Working Draft, W3C (Nov 26 2019), https://www.w3.org/TR/2019/WD-dx-prof-conneg-20191126/

[162] Svensson, L.G., Atkinson, R., Car, N.J.: Content Negotiation by Profile Implementation Report. W3C-internal document, W3C (Mar 21 2022), https://w3c.github.io/dx-connegp/connegp-implementation-report/

[163] Svensson, L.G., Verborgh, R., Van de Sompel, H.: Indicating, Discovering, Negotiating, and Writing Profiled Representations. Internet-draft, Internet Engineering Task Force (Mar 9 2021), https://www.ietf.org/archive/id/draft-svensson-profiled-representations-01.txt

[164] Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Négociation de Contenu sur le Web: un État de l'Art. In: Journées Francophones d'Ingénierie des Connaissances (IC 2022) @ Plate-Forme Intelligence Artificielle (PFIA 2022) (2022), https://hal.science/hal-03837655v1/file/Actes-IC-PFIA-Taghzouti-2022.pdf

[165] Taghzouti, Y.: Enable Decentralised Semantic Content Negotiation through Equivalence Links. In: SeReCo Summer Workshop 2023. Waischenfeld, Germany (Jul 2023), https://hal-emse.ccsd.cnrs.fr/emse-04138801

[166] Taghzouti, Y., Vachtsevanou, D., Mayer, S., Ciortea, A.: A Step Toward Semantic Content Negotiation. In: Companion Proceedings of the 23rd International Conference on Knowledge Engineering and Knowledge Management, Bozen-Bolzano, Italy, September 26-29, 2022. CEUR Workshop Proceedings, vol. 3256. CEUR-WS.org (Sep 2022), https://ceur-ws.org/Vol-3256/paper5.pdf

[167] Taghzouti, Y., Zimmermann, A., Lefrançois, M.: A Portal for the State of the Art on Content Negotiation. In: Proceedings of the ISWC 2022 Posters, Demos and Industry Tracks: From Novel Ideas to Industrial Practice co-located with 21st International Semantic Web Conference (ISWC 2022), Virtual Conference, Hangzhou, China, October 23-27, 2022. CEUR Workshop Proceedings, vol. 3254. CEUR-WS.org (Oct 2022), https://ceur-ws.org/Vol-3254/paper362.pdf

[168] Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Négociation de contenu sémantique pour l'échange de connaissances entre systèmes hétérogènes. In: 20èmes Rencontres des Jeunes Chercheurs en Intelligence Artificielle. RJCIA Rencontres des Jeunes Chercheurs en Intelligence Artificielle PFIA 2022 (Jun 2022), https://hal.science/hal-03765368

[169] Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Content Negotiation in a Decentralised Semantic Context Utilising Equivalence Links. In: Joint Proceedings of the ESWC 2023 Workshops and Tutorials co-located with 20th European Semantic Web Conference (ESWC 2023), Hersonissos, Greece, May

28-29, 2023. CEUR Workshop Proceedings, vol. 3443. CEUR-WS.org (2023), https://ceur-ws.org/Vol-3443/ESWC_2023_DMKG_paper_9797.pdf

[170] Taghzouti, Y., Zimmermann, A., Lefrançois, M.: Content Negotiation on the Web: State of the Art. Tech. Rep. abs/2204.10097, CoRR (Apr 21 2022), https://arxiv.org/abs/2204.10097

[171] Tandy, J., van den Brink, L., Barnaghi, P.: Spatial Data on the Web Best Practices. W3C Working Group Note, W3C (Sep 28 2017), https://www.w3.org/TR/sdw-bp/

[172] Taylor, M., Weiss, Y.: User-Agent Client Hints. W3C Commu Group Draft Report, W3C (Sep 8 2023), https://wicg.github.io/ua-client-hints/

[173] Tennison, J., Reynolds, D., Dodds, L.: Linked Data API Vocabulary (api) (Feb 19 2010), https://lov.linkeddata.es/dataset/lov/vocabs/api

[174] Thapa, R.B., Giese, M.: Mapping Relational Database Constraints to SHACL. In: The Semantic Web - ISWC 2022 - 21st International Semantic Web Conference, Virtual Event, October 23-27, 2022, Proceedings. Lecture Notes in Computer Science, vol. 13489, pp. 214–230. Springer (Oct 2022), https://doi.org/10.1007/978-3-031-19433-7_13

[175] Verborgh, R.: Your JSON Is Not My JSON – A Case for More Fine-Grained Content Negotiation. In: Proceedings of the Workshop on Smart Descriptions & Smarter Vocabularies (Nov 2016), https://ruben.verborgh.org/articles/fine-grained-content-negotiation/

[176] Verborgh, R., Sande, M.V., Shankar, H., Balakireva, L., de Sompel, H.V.: Devising affordable and functional linked data archives. Bull. IEEE Tech. Comm. Digit. Libr. **13**(1) (2017), https://bulletin.jcdl.org/Bulletin/v13n1/papers/verborgh.pdf

[177] Verstak, A.: Google Scholar (2004), https://scholar.google.com/

[178] Viot, P., Thouvenin, N.: ISTEX: Une Nouvelle Corde à Son ARK. Arabesques **88**, 18–19 (2018), https://publications-prairial.fr/arabesques/index.php?id=1222

[179] van der Vlist, E.: RELAX NG - A Simpler Schema Language for XML. O'Reilly (2004), http://www.oreilly.de/catalog/relax/index.html

[180] Van der Vlist, E.: Schematron. O'Reilly Media, Inc (2007), https://meta.geo.census.gov/data/existing/decennial/GEO/GSCQB/ReleasedPerlScriptsandSchemas/XSLTSSchemsaModules/schematron/schematron/Schematron.pdf

[181] Weyns, D., Omicini, A., Odell, J.: Environment as a First Class Abstraction in Multiagent Systems. Auton. Agents Multi Agent Syst. **14**(1), 5–30 (2007), https://doi.org/10.1007/s10458-006-0012-0

[182] Wilde, E.: The 'profile' Link Relation Type. RFC 6906, Independent Submission (Mar 2013), https://doi.org/10.17487/RFC6906

[183] Winstanley, P., Coyle, K.: Dataset Exchange Working Group Charter. W3C Working Group Charter, W3C (May 4 2017), https://www.w3.org/2017/dxwg/charter

[184] Wooldridge, M., Jennings, N.R.: Intelligent Agents: Theory and practice. The knowledge engineering review **10**(2), 115–152 (1995), https://core.ac.uk/download/pdf/1498750.pdf

[185] Wright, A., Andrews, H., Hutton, B., Dennis, G.: JSON Schema: A Media Type for Describing JSON Documents (Jun 16 2022), https://json-schema.org/draft/2020-12/json-schema-core

[186] Wright, A., Andrews, H., Hutton, B., Dennis, G.: JSON Schema: A Media Type for Describing JSON Documents. Internet-Draft draft-bhutton-json-schema-01, Internet Engineering Task Force (Jun 10 2022), https://datatracker.ietf.org/doc/draft-bhutton-json-schema/01/, Work in Progress

[187] DOI Content Negotiation, https://citation.crosscite.org/docs.html

[188] Geoscape - The Predictive Address Verification API, https://docs.geoscape.com.au/docs/oas-predictive-api/

[189] ApacheWeek - Content Negotiation Explained (Jul 26 1996), http://www.apacheweek.com/features/negotiation

[190] Restify Documentation - Versioned Routes (Dec 7 2018), http://restify.com/docs/home/#versioned-routes

[191] Apache - Content Negotiation (2020), https://httpd.apache.org/docs/2.4/content-negotiation.html

[192] Schema.org Vocabulary V7.0 (Mar 10 2020), https://lov.linkeddata.es/dataset/lov/vocabs/schema

[193] Content Negotiation in ASP.NET Web API (Nov 5 2022), https://learn.microsoft.com/en-us/aspnet/web-api/overview/formats-and-model-binding/content-negotiation

[194] Django Rest Framework - API Guide - Content negotiation (Oct 12 2022), https://www.django-rest-framework.org/api-guide/content-negotiation/

[195] PlayFramework - Documentation - Content Negotiation (Oct 20 2022), https://www.playframework.com/documentation/2.8.x/ScalaContentNegotiation

[196] Spring MVC Content Negotiation (2022), https://www.baeldung.com/spring-mvc-content-negotiation-json-xml

[197] API Platform - Content Negotiation (2023), https://api-platform.com/docs/core/content-negotiation/

[198] DataCite Content Negotiation (May 2023), https://support.datacite.org/docs/datacite-content-resolver

# Résumé

Le Web fournit un accès à un vaste ensemble de ressources interconnectées. Ces ressources peuvent avoir plusieurs représentations qui reflètent leurs états et varient selon une ou plusieurs dimensions. Pour cela, les agents web entreprennent une phase de négociation pour sélectionner la plus appropriée. La dimension du profil est celle qui nous intéresse, où un profil est une description des contraintes sémantiques et/ou structurelles d'une représentation souhaitée.

L'objectif de cette thèse est de défendre l'intérêt d'une négociation de contenu fine et souple en tenant compte du fait que la négociation de contenu n'est pas un processus monolithique, mais plutôt composé de plusieurs étapes. Tout d'abord, nous proposons un modèle abstrait pour la négociation de contenu que nous appelons CON-NEG. Ensuite, nous présentons les améliorations que nous avons apportées à chaque étape du processus: découverte de représentations potentielles grâce à des liens d'équivalence; indication de contraintes plus fines grâce à leur sévérité; relaxation de la validation; rapport d'une validation plus fine; interprétation du rapport de validation. Ensuite, nous présentons le modèle abstrait S-CON-NEG, une extension pour la négociation fine et relaxée du contenu. Enfin, nous proposons une typologie de modèles plus fins pour la négociation de contenu dans la dimension de profil. Une instanciation du modèle abstrait est ensuite proposée avec des expériences pour démontrer l'applicabilité et les avantages de nos méthodes. Aussi, nous illustrons comment faciliter l'interaction des systèmes hétérogènes en appliquant notre approche à un cas d'utilisation de systèmes multi-agents sur le Web.

Mots clés : Négociation de contenu, Web sémantique, HTTP, validation sémantique

# Abstract

The World Wide Web started as an information management initiative with the aim of providing universal access to a vast set of interconnected resources. Resources in this space could have multiple representations that capture their state and vary along some dimensions. In such a case, web agents undertake a negotiation phase to select an appropriate representation.

Besides, the Semantic Web adds structure and meaning to web content, making it a suitable area for collaboration between human and software agents. Concretely, resource representations are described using the Resource Description Framework by means of various ontologies and vocabularies. This has motivated the investigation of content negotiation in the profile dimension, which is orthogonal to the previous dimensions, where a profile is a description of semantic and/or structural constraints of a desired representation.

The aim of this thesis is to argue for the worth of a flexible (e.g. fine-grained constraints) and relaxed (e.g. in case of non-conformance of optional constraints) content negotiation taking into account that content negotiation is not a monolithic process, but rather composed of several stages: (1) discovery; (2) request formulation; (3) selection and/or adaptation; (4) response indication; (5) response interpretation.

First, we propose an abstract model for content negotiation we call CON-NEG (for CONtent NEGotiation). Then, we present the enhancements we made at each stage for the process: (1) discovering potential representations through equivalence links; (2) indicating finer constraints through severity; (3) relaxing validation; (4) reporting finer validation; (5) interpreting the validation report. After that, we introduce S-CON-NEG abstract model, for Semantic CON-NEG, which is its extension for flexible and relaxed content negotiation. Finally, we propose a typology of finer patterns for content negotiation in the studied profile dimension.

An instantiation of the abstract model is later proposed along with experiments to demonstrate the applicability and benefits of our methods. In addition, we illustrate how such improvements towards facilitating heterogeneous systems interaction by applying our approach in a Web-based multi-agent systems use case.

Keywords: Content Negotiation, Semantic Web, HTTP, Semantic Validation.