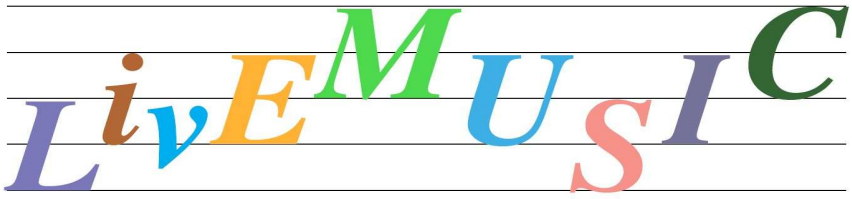

Project's Final Report

Living Environmnent Monitoring Use Scenario with Intelligent Control

Project Description and Results

Hassan Aït-Kaci, Sofiene Chamakhi, Tanguy Raynaud, Mathieu Veyrand

January 2016

Publication Note

Authors' address:

LIRIS—UFR d'Informatique
Université Claude Bernard Lyon 1
43, boulevard du 11 Novembre 1918
69622 Villeurbanne cedex
France

Phone: +33 (0)4 27 46 57 08
Email: livemusic@univ-lyon1.fr

This work was carried out as part of the *LivEMUSIC* Project (*Living Environmnent Monitoring Use Scenario with Intelligent Control*) funded by the *Programme Avenir Lyon Saint-Etienne* (PALSE) of the *Université de Lyon*'s « *Investisssements d'Avenir* » (**ANR-11-IDEX-0007**) at the *Université Claude Bernard Lyon 1*.

Project's Final Report

Living Environmnent Monitoring Use Scenario with Intelligent Control

Project Description and Results

Hassan Aït-Kaci, Sofiene Chamakhi, Tanguy Raynaud, Mathieu Veyrand

livemusic@liris.cnrs.fr

January 2016

Abstract

This document describes the *LivEMUSIC* project (*Living Environmnent Monitoring Use Scenario with Intelligent Control*), its activities and results.

Keywords: Urban Milieu; Intelligent Monitoring; Machine Learning; Knowledge Extraction; Knowledge-Base Querying and Reasoning

Résumé

Ce document décrit le projet *LivEMUSIC* (*Living Environmnent Monitoring Use Scenario with Intelligent Control*), ses activités et résultats.

Mots-Clés: Milieux urbain; supervision intelligente; apprentissage automatique; extraction de connaissances; traitement des requêtes et raisonnement à base de connaissances

Table of Contents

1 Introduction and Project Overview

This document reports work done in the course of the $\mathcal{L}ivE\mathcal{MUSIC}$ Project,[1] a one-year research-application experimental effort carried out at the LIRIS under the supervision of Prof. Hassan Aït-Kaci. This project was a part of the "*Projet Avenir Lyon Saint-Etienne*" (PALSE),[2] under the theme "*Intelligence des Mondes Urbains*" (IMU).[3]

$\mathcal{L}ivE\mathcal{MUSIC}$ stands for "Living Environment Monitoring Use Scenarios with Intelligent Control." Its objective is to experiment on real data from real-time monitoring, extracting knowledge from these data, and leveraging this knowledge to process them in use scenarios making "intelligent use" of this information. By "*intelligent use*," we mean being able to derive implicit information, or information requiring some form of data analysis.

1.1 Summary

The project has consisted in experimenting with plausible use scenarios in intelligent living-environment monitoring using data collected in the Grand Lyon. The technology demonstrated takes on learning from raw data, knowledge representation, and automated reasoning, as means to enable "smart" use of some aspects of living environment from raw data, focusing on urban and social milieu.

The project's main motivation has been the opportunity of actual data being available from the Grand Lyon and made accessible for our experimentation by the project sponsors—*Université de Lyon* (UdL) and *Universté Claude Bernard Lyon 1* (UCBL). It is thus only a small part among many of the Grand Lyon's collection of scientific efforts researching and experimenting with AI-based R&D software supporting "Smart Cities" applications.

1.2 Context

All aspects of life are being overseen, monitored, and controlled through hybrid Perception/-Analysis/Control (PAC) networks. Software-enabled technology prevails as the ubiquitous glue bringing all aspects of such monitoring to bear on its inner workings. However, human actors bring most, if not all, of the intelligence of such software. With the increased networking of all aspects of society, software intelligence has made great strides in showing a capacity for increased autonomy in the distributed monitoring of PAC networks—the most critical part being the "A" (Analysis).

This project lies precisely in this line of technologies that attempt to meet the needs for the development of an intelligent monitoring framework that can provide policy makers and analysts with high added-value information. In this context, the project has experimented with combining specific, more semantic-based, techniques for distributed knowledge and information handling developed in areas such as knowledge representation and reasoning, and intelligent query processing. To give a simple example to illustrate, such could consist in replacing a thermal control system that abruptly adjusts the ambient temperature to a higher temperature as the weather chills, to one that

[1] https://liris.cnrs.fr/.../un-chercheur-du-liris-laureat-de-lappel-package-palse
[2] http://palse.universite-lyon.fr/
[3] http://imu.universite-lyon.fr/

does so gradually and preventively by understanding the trend of weather from local data such as temperature, pressure, and precipitations, as well as from the analysis of global networked weather measurements and reports.

Thus, our research context concerns several critical aspects where unified access to distributed information for decentralized monitoring is of central importance, such as: enriching raw data extracted from measurements to enable conflict resolution and easy interpretation; aligning these enriched data with formal models of information quality; representing the meaning of this information with the use of ontologies to support semantic understanding across various domains (*e.g.*, transportation, health, education, *etc.*, . . .).

1.3 Objective

As its title indicates, the $\mathcal{L}iv\mathcal{EMUSIC}$ project's plan is an illustration in the form of a use scenario of a particular experiment using real data. Data are used to derive ontological knowledge. Knowledge is extracted from data and leveraged in processing these data in application scenarios, thus making "intelligent" use of this information. Our specific experiment has focused on using taxonomic knowledge for reasoning in a simple but efficient order-sorted propositional logic. In other words, the "reasoning" consists in processing (representations of) concepts and subconcepts composed with the Boolean operations and, or, not. Such conceptual taxonomies are first derived in a knowledge extraction phase relying on Formal Concept Analysis [19], while the reasoning relies on a prototype tool: the $\mathcal{CEDAR}$ reasoner for attributed ontologies [3, 6, 1].[4]

We took advantage of an agreement between the LIRIS[5] and the Grand Lyon[6] giving access to over 500 real data in various applications.[7]

1.4 Organization

Basics

- **Nature:** Projet Avenir Lyon Saint-Etienne (PALSE), Université de Lyon
- **Duration:** one year (January 15, 2015–January 14, 2016)
- **Host Institution:** LIRIS, Université Claude Bernard Lyon 1
- **Name:** $\mathcal{L}iv\mathcal{EMUSIC}$—*$\mathcal{L}iv$ing $\mathcal{E}$nvironmnent $\mathcal{M}$onitoring $\mathcal{U}$se $\mathcal{S}$cenarios with $\mathcal{I}$ntelligent $\mathcal{C}$ontrol*
- **Objective:** Design and prototype a software system capable of extracting implicit ontological structures from raw monitoring data and apply it in a use case example in which such inferred knowledge can provide intelligent assistance.

[4] http://cedar.liris.cnrs.fr/
[5] https://liris.cnrs.fr/
[6] http://data.grandlyon.com/
[7] http://data.grandlyon.com/acceder-aux-donnees/

Staffing

- **Full-time member:**
  - Prof. Hassan Aït-Kaci, project leader (Jan. 2015–Jan. 2016)
- **Part-time members:**
  - Tanguy Raynaud, part-time engineer (Feb. 2015–Apr. 2015)
  - Sofiene Chamakhi, M2 intern (Feb. 2015–Jun. 2015)
  - Mathieu Veyrand, part-time engineer (May 2015–Oct. 2015)

We also gratefully acknowledge Prof. Marian Scuturici's advising regarding access to the Data Grand Lyon platform in Task T1, as well as Dr. Samir Amir's help providing technical assistance for Task T3's use of the $\mathcal{CEDAR}$ reasoner.

2 Project's Planned Schedule and Tasks

Figure 1 depicts the overall structure of the $\mathcal{L}ivE\mathcal{MUSIC}$ project's schedule and task breakdown. The project comprises four tasks, each described below.

- **Task T1**
  The objective of this task is to inspect and sort out massive raw data gathered from primary sources of all kinds, such as sensor devices, surveillance videos, manually entered inputs, *etc.*, ensuring that it is usable for further processing in a consistently uniform RDF representation. Part of the raw data in basic RDF form to be thus selected out are meant to comprise triplestores to be further processed into ontological knowledge to be extracted.
- **Task T2**
  This task concerns the processing of RDF triplestores generated from raw sensor data in order to derive ontological schemas that the generated RDF triples abide by. The objective is to categorize these triplestore data using classes and attributes fitting their structure using Formal Concept Analysis ($\mathcal{FCA}$). The outcome is to constitute an inferred ontology for these data that can be used for reasoning about such data.
- **Task T3**
  This task is to develop a use case exploiting ontological terminological knowledge (TBox) over existing monitoring data (ABox) with semantic reasoning based on $\mathcal{OSF}$-constraint logic to optimize information retrieval, data analytics, and query processing. The object is to demonstrate how to leverage the capabilities brought by knowledge-based processing of monitoring data for intelligent applications with plausible use case scenarios exploiting actual gathered data.
- **Task T4**
  This task is the final reporting of the work carried out during the project—the packaging of the software and preparation and delivery of technical reports, including this final one.

The rest of this document develops the activities and results for each task.

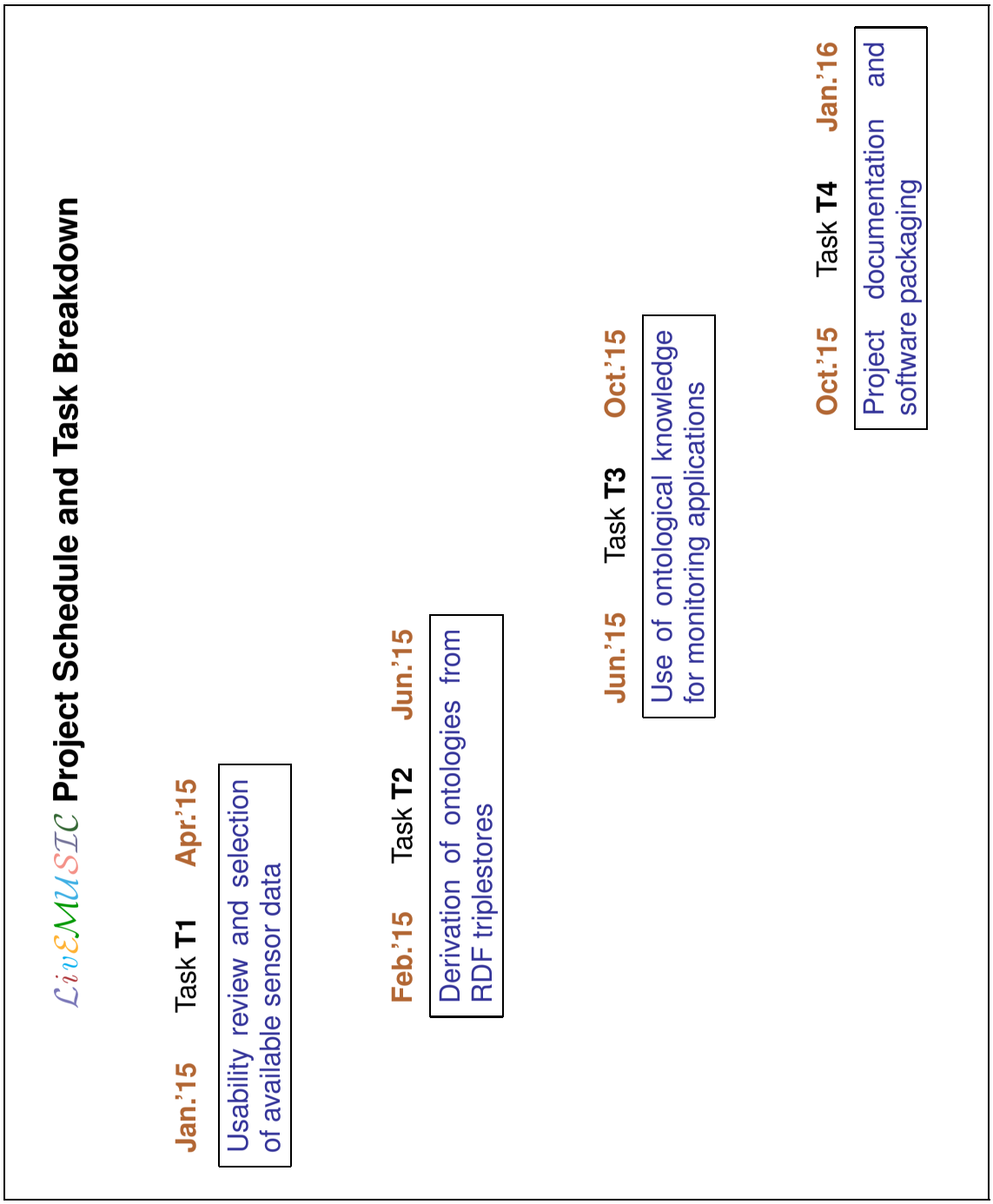

Figure 1: The *LivEMUSIC* Project Structure

3 Task T1: Usability Review and Selection of Available Sensor Data

This task was assigned to Tanguy Raynaud under Hassan Aït-Kaci's supervision (Mid-February–Mid-April 2015).

3.1 Task objective and rationale

It is often overlooked that, while scientific research relies on formal theoretical models, an important (indeed, vital!) part of the work concerns access to and use of real data. The LivEMUSIC's very motivation actually arose from the unique opportunity of access to such data from the Grand Lyon.[8]

Nowadays, there exist a multitude of sources of raw data providing all sorts of records and measurements respecting no specific terminological schema. Such sources gather data in primitive forms for no specific intended purpose other than keeping a record. Such forms do not follow any particular terminological classification schema. For example, in the public data made available by the Lyon metropolis, one can find information resulting from sensors and measurements in plain text or primitive numerical forms. Such records do not anticipate any specific use.

Therefore, an important initial phase for such an experimental project as LivEMUSIC needed our initial attention. The task was not so much a technological one but one requiring taking a look at a large set of potential raw data collection that could be accessible to us from the Grand Lyon family of databases, and make a choice for a plausible use scenario where reasoning over data can be effectively used for a helpful application.

The choice had to satisfy certain criteria to meet our time and tooling constraints, much of it necessitated for further processing. These criteria were essentially:

- to conceive a plausible use case that could be designed to exploit actual data from the Grand Lyon for yet unprovided similar services thanks to "intelligent" processing of such data;
- to develop experimental software demonstrating this use scenario while being able to learn from raw data and reason to draw helpful conclusions from knowledge *implicit* in these data, in addition to the usual services rendered by standard DB system processing of its *explicit* values.

Before using any ontological knowledge, one must first acquire such knowledge. Indeed, before reasoning one must learn the concepts forming one's knowledge base. Hence, an important attributed-concept learning phase for deriving a conceptual ontology from fielded data records was Task T2's purpose, as described in Section 4. But even prior to such learning was the necessity to identify what might be a usable dataset from which to derive ontological knowledge in the large collection made available as Smart Data Grand Lyon.[9] This was essentially Task T1's motivation and purpose.

More specifically, Task T1's dataset assessement consisted in:

- taking a detailed look at about 300 datasets among the 576 available;

[8] http://data.grandlyon.com/acceder-aux-donnees/
[9] http://data.grandlyon.com/

- selecting of a few potential candidates;
- making a final choice of $\mathcal{L}iv\mathcal{EMUSIC}$'s experimental dataset.

3.2 Selection of exploitable datasets

A huge amount of data, of various types, and from various sources has been made available by the Grand Lyon in the form of raw (essentially text-valued) fielded-relation databases for download from their website.[10] Most, but not all, of these data are freely available. A special agreement between the Grand Lyon and the LIRIS made the entirety of these databases accessible to the $\mathcal{L}iv\mathcal{EMUSIC}$ project. At present, there are over five hundreds databases, containing about tens of thousands of attributes. In total, these databases contain millions of record entries in multiple domains and applications, making this data bank the richest and most comprehensive data source available for this region. The contents of these databases are quite varied, containing data from sensors, manual inputs, or external sources. The provided datasets are categorized into 14 activity areas.[11]

Some of the data are static (maps, places, names, *etc.*, ...). But most of the data are dynamic. They are "live" in that they are snapshots taken at regular time intervals of the current state of a given set of static data. Each snapshot is updating a stream record of the evolution of a database. Obviously, the update rate is quite different depending on the nature and source of the data. Some databases are updated only once or twice a year, particularly in the case of geographical databases, while some others every months, weeks, days, hours, or even in real time for some databases such as available transportation, or current traffic analysis.

Clearly, a detailed assessment of *all* these datasets was beyond feasibility. This is because of the enormous quantity and variety of datasets to be reviewed, as well as the relatively limited time and human resource available in the $\mathcal{L}iv\mathcal{EMUSIC}$ project. Another reason is that Task T1's purpose was to provide a workable set of potential data from which Tasks T2 and T3 could actually experiment with. The use scenario developed in Tasks T2 and T3 can thus be seen as a generic experiment with some actual Grand Lyon data for intelligent processing (learning, reasoning, querying). Task T1 was to make a rational dataset classification and eventual selection being tributary to the intended use to be made of the data.

Pragmatically, the $\mathcal{L}iv\mathcal{EMUSIC}$ project was to prototype and demonstrate some realistic use case providing a means to derive useful information from actual raw data upon a user's request expressed as abstract specifications. Thus, the data made available by the Grand-Lyon were ideal for this kind of scenario. An important reason for this is because the data entries are authentic and dynamic—some of it even real-time. However, the notion of "real-time" here must be clarified. We mean *synchronous* real time. Indeed, for the sake of coherent processing despite potential latencies, data are updated as they evolve at regular time intervals between updates. We decided for this study that all databases with an update rate less than one hour could be considered as real-time databases. Thus, an initial choice of a few potential databases was made. The selected sets included public transportation, airports schedules, traffic sensors, roadwork, *etc.*, ...

Besides real-time databases, we also selected other databases, including geographical ones or in-

[10] http://data.grandlyon.com/

[11] In French: *transports, imagerie satellite, citoyenneté, services, culture, localisation, limites administratives, économie, environnement, occupation du sol, urbanisme, équipements, accessiblilité, démographie*.

volving important places of Lyon. This was in order to diversify the choice potential for alternative scenario contexts in case we encountered unforeseen problems in our selection of a template exemplar. Thus, a few other datasets along real-time ones were added in the selection. They included public transportation stations, city streets, car park, and Velo'V locations. The main criteria used to select the additional databases in this step were the presence of at least one attribute that could be used to link with the real-time database. These attributes were mainly geographical, like the addresses, towns, or any location information.

The final selection was made arbitrarily based on the wealth of information contained in the databases. It was necessary to find a compromise between an exploitable number of databases, and contents that could lend itself to semantic reasoning of some sort. That resulted in a total of 38 datasets out of the 576 available from the Grand Lyon. In Appendix Section A, we give a list of exploitable datasets to be the ones we preselected in the course of Task T1.

A few areas of the Grand Lyon databases, including car park, Velo'V, and public transportations, offer many datasets for a single entity. These entities are often divided in two datasets: one static and the other dynamic. The static dataset is for permanent attributes like location, capacity, such unchanging information. The dynamic dataset is for real-time attributes, such as available spots, current status, *etc.*, ... This setup offers the advantage to ease their being used jointly in relatively complex queries across datasets.

The foregoing rationale eventually identified the data provided for Velo'V as a suitable candidate for the LivEMUSIC project.

4 Task T2: Derivation of Ontologies from RDF Triplestores

This task consisted in the Master-2 internship work assigned to Sofiene Chamakhi under Hassan Aït-Kaci's supervision (Mid-January–Mid-June 2015) [10]. This section is an English version of relevant parts of Sofiene Chamakhi's MS report, translated, adapted, and edited by Hassan Aït-Kaci.

Task T2 concerns the analysis of raw RDF data in order to classify it according to specific properties and derive terminological schemas that the so-classified generated RDF triples abide by. The idea is to categorize the raw data using classes and attributes fitting their structure inferred using Formal Concept Analysis ($\mathcal{FCA}$). The outcome of such an analysis can then be used as terminological knowledge for reasoning about these data classified based on implicit properties it verifies. This categorization provides approximations whereby similar data are semantically aggregated by conceptual scaling, in effect giving the power of a "semantic zoom" determining the level of detail at which one wishes to cluster raw data together into approximation classes.

4.1 Context and problem definition

There exist some techniques for deriving ontological knowledge structure implicit in raw data. We investigated some of these approaches among the most adequate with aim to adapt any that could serve our specific needs under the requirements of a 4-month master-level research project using the public data made accessible by the Grand Lyon. The objective was to extract intensional definitions in the form of implicit logical properties characterizing the available data. The idea

was to put the data in a more "semantical" form enabling using it in useful applications. Among the approaches we reviewed for this purpose, we opted for Formal Concept Analysis ($\mathcal{FCA}$) [19]. We will justify our choice of method, as well as the selection of the $\mathcal{FCA}$ software tool we made use of after an overview of those we considered.

The rest of this section is organized as follows. Section 4.2 justifies our approach, giving the essential background on $\mathcal{FCA}$, the analytical method we used. Section 4.3 describes a few specific applications where deriving implicit ontologies based on $\mathcal{FCA}$ is leveraged, then reviews several software tools that are based on $\mathcal{FCA}$, and explains how we proceeded to make a rational choice of which tool to use. Section 4.4 details how we used $\mathcal{FCA}$ in $\mathcal{L}iv\mathcal{E}\mathcal{M}\mathcal{U}\mathcal{S}\mathcal{I}\mathcal{C}$: the general architecture of the system we implemented for our approach, how we used these techniques on the specific dataset we selected among those provided by the Grand Lyon.

4.2 Justification of our approach

Constraints and requirements

For the sake of motivating our choice of tools for carrying out the work required in Task T2 within the time constraints and academic context of the project,[12] in this section we proceed to a rapid overview of the state of the art we could find available to us. For that phase of Task T2, the issue was not to carry out a complete in-depth review of the field of ontology learning. Rather, we were driven by the pragmatics of fulfilling the following requirements:

- ensuring the availability of an open-source tool for our use;
- ensuring the adequacy of such a tool for our purposes.

In addition, this task involving technical material of a nature previously unfamiliar to the task manager, it was important to delegate to him the choice of the tool on one that he would feel most confident with. Thus, this section justifies the process that led to the eventual choice of tool to be used for this task.

Formal concept analysis

Formal Concept Analysis ($\mathcal{FCA}$) is a mathematical method that was originally introduced by Rudolf Wille [19]. It is used for deriving an implicit conceptual ordering from attributed data based on the lattice-theoretic notion of Galois Connection [18]. With $\mathcal{FCA}$, it is possible to analyze raw attributed data and find which attributes are common to which types of objects, and dually which objects are common to which types of attributes.

More precisely, $\mathcal{FCA}$ uses the notion of *formal context*, which is defined as a set of triples of the form $\langle O, A, R\rangle$, where O is a set of objects, A is a set of attributes, and R is a binary relation between the objects and the attributes—*i.e.*, $R \subseteq O \times A$. A formal concept in such a context is defined as a maximal set of objects in O possessing the same set of attributes in A, or equivalently, as a maximal set of attributes possessed by the same set of concepts.

[12] Task T2's task manager was Sofiene Chamakhi as it comprised his internship project for the obtention of the degree of Master 2 in Computer Science from the UCBL.

For example, as shown in Figure 2, if the objects (denoting persons in this case: Jean, Michel, Marie, Julie) listed in the leftmost column possess the attributes (each person's gender and interests in this case: Homme, Femme, Sport, Voyages, Voitures, Mode) listed in the top row as given by the checkmarks in each corresponding row, $\mathcal{FCA}$ can derive the object-attribute set lattice shown in Figure 3.

	Homme	Femme	Voitures	Sport	Mode	Voyages
Julie		✓			✓	
Jean	✓					✓
Marie		✓		✓		
Michel	✓		✓	✓		

Figure 2: Example of a formal context

Each node in this lattice is a pair of sets: a set of objects (here, persons—in green) and a set of attributes (here, a person's gender or interests—in red), such that all the objects of the pair possess all the attributes of the pair (and dually all the attributes of the pair are possessed by all the objects of the pair). Thus, going up the lattice for object sets is by set union and going down is by set intersection (*i.e.*, the object-set ordering in set inclusion). Therefore, the topmost set of objects is the set of all the objects, and the bottommost set of objects is the empty set. Dually, going up the lattice for attribute sets is by set intersection, and going down by set union. Therefore, the topmost set of attributes is the empty set, and the bottommost set of attributes is the set of all attributes.

This lattice is also depicted as Figure 4 with more concise labelling of the nodes.[13] It is the same lattice as the object-attribute set lattice in Figure 3 where a node represents a formal concept derived from the formal context specified in Figure 2. The only difference is in the labelling of nodes. An object is indicated as a rectangular box with white background, whereas an attribute is indicated on a greyed rectangular box. In this concise form, an object possesses all the attributes above it. Dually, an attribute is possessed by all the objects below it.

4.3 $\mathcal{FCA}$ in practice

Examples of applications leveraging $\mathcal{FCA}$

Extracting implicit knowledge using $\mathcal{FCA}$ and extensions has been applied in many venues and situations. We next give a brief *aperçu* of such examples as illustrations.

Despite its simplicity and usefulness, $\mathcal{FCA}$ as originally formulated (see, *e.g.*, [19]) and as described above suffers from potentially generating concept lattices of enormous size as the number

[13] This form is the one generated by the Lattice Miner FCA tool—see Section 4.3, Page 12.

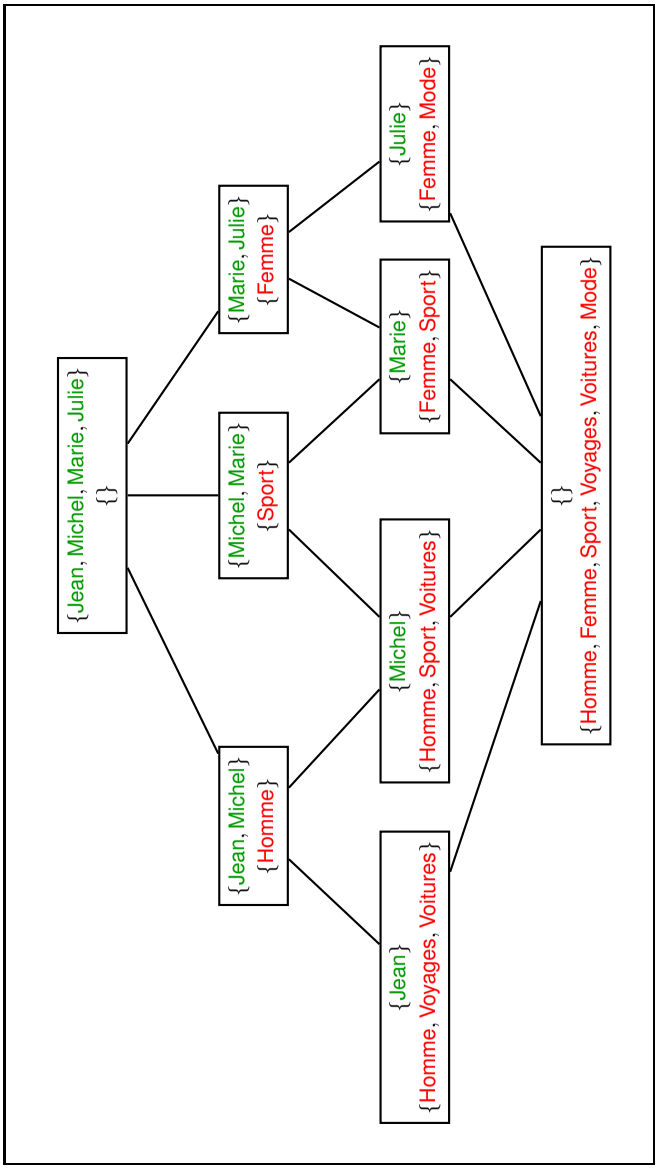

Figure 3: The object-attribute set concept lattice for the formal context of Figure 2

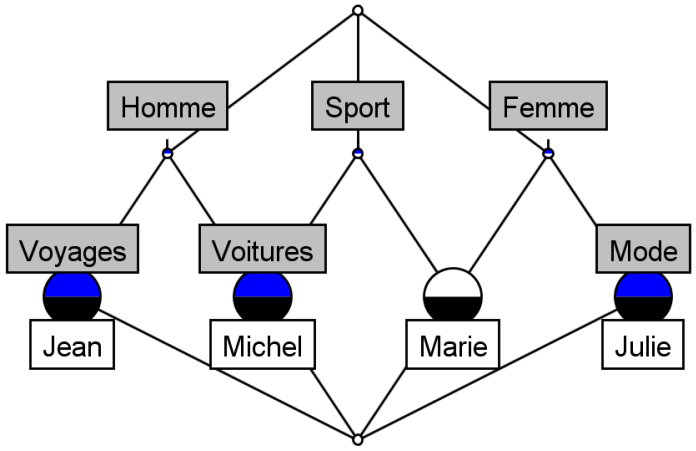

Figure 4: Concept lattice of Figure 2 rendered with Lattice Miner using concise labels

of objects and attributes grows. This is because one considers only binary (yes/no) values in a formal context.

In [16], it is shown how to adapt $\mathcal{FCA}$ to work with the RDF data format as well as using $\mathcal{FCA}$ on multi-valued formal contexts. Such extensions offer an important added functionality to $\mathcal{FCA}$ as it enables working with values of any data type (character strings, numbers, dates, locations, *etc.*). A serendipitous consequence of this allows grouping together subsets of attributes as pertaining to a specific type of object, thereby reducing the size of the context and enabling $\mathcal{FCA}$ processing to scale up. Such a technique, dubbed "*conceptual scaling*," is a preprocessing step the authors propose to use when dealing with multi-valued contexts. It consists in defining threshold values allowing to group or dissociate attributes. Such a multi-valued context can then be converted into a binary one, from which a smaller lattice can then be extracted. For example, on the left of Figure 5, one can see a conventional concept lattice where objects and attributes are integers. Each number object possesses the attribute denoted by same number (*i.e.*, object number n possesses attribute number n, for $n = 1, 2, \ldots$). The lattice on the right of the figure is obtained after conceptual scaling of the one on the left by grouping attributes with values greater or less than 3. Clearly, the number of concepts in this lattice is greatly reduced.

Some have investigated applying $\mathcal{FCA}$ for formal concept discovery from Semantic Web data where datasets have the particularity of being very large. For example, the work reported in [21], looks at using $\mathcal{FCA}$ on Linked Open Data [9]. Other approaches consider the same challenge (deriving ontologies from data resulting from $\mathcal{SPARQL}$ queries) exploiting properties of specific application contexts. Thus, in [11] and [5] $\mathcal{FCA}$ is used to extract knowledge from analysis of responses to $\mathcal{SPARQL}$ queries. The objective in so doing is to try to organize and classify the data retrieved by $\mathcal{SPARQL}$ queries into hierarchies, a challenge when dealing with conventional data formats such as XML/RDF, JSON, CSV, HTML, *etc.*, ...

$\mathcal{FCA}$ has also proved useful for ontology alignment [28, 15] where it enables discovering new concepts from those of different ontologies in which they did not exist explicitly. This information,

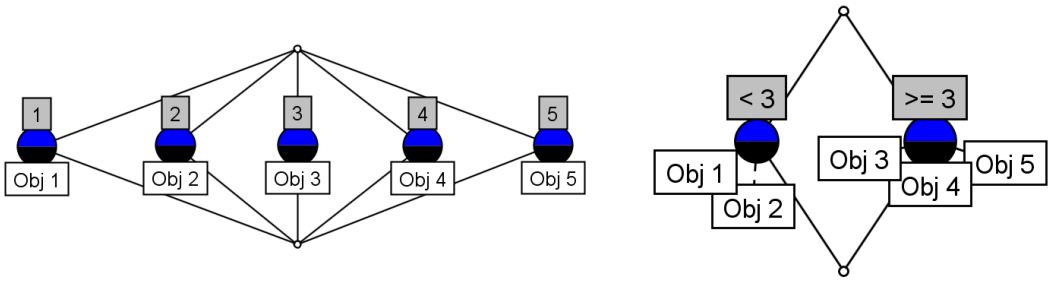

Figure 5: Example of conceptual scaling

then, can enable linking two ontologies sharing similar objects or attributes by making $\mathcal{FCA}$ reveal common implicit concepts. It can as well be used to eliminate redundant concepts.

Making operational use of $\mathcal{FCA}$-derived ontologies in deductive reasoning has been experimented through logical interpretations. For example, in [20], it is proposed to translate an $\mathcal{FCA}$-induced ontology into statements that are First-Order Logic Horn clauses, which can then be interpreted operationally as Prolog programs.

Applications to natural-language text processing and understanding have been carried out using $\mathcal{FCA}$. For example, in [25], it is shown how $\mathcal{FCA}$ can be used for building linguistic ontologies as well as for comparing and aligning lexical databases. The advantage is that, when these databases are expressed using conceptual lattices, the knowledge extracted is in effect language-independent. Thus, the derived ontologies can serve as semantic references to manage lexical data across multiple languages .

$\mathcal{FCA}$ has also been used for constructing ontologies not from data, but from text [12]. The method relies on resolving parsing ambiguities while processing text by tagging each word with possible transition probabilities, giving eventually decision tree tagging a sentence estimating its probable structure from the transition probabilities of its constituents (*e.g.*, TreeTagger [26], LoPar [27]). Pairs of likely transitions among syntactic constituents, such as $\langle subject, verb\rangle$, $\langle verb, object\rangle$, or $\langle verb, proposition\rangle$, are extracted and put in lexical normal form. Then, $\mathcal{FCA}$ is used on the probabilistic trees to derive ontologies of categories of lexical and syntaxic constructs.

The foregoing rapid tour of some illustrations of techniques using $\mathcal{FCA}$ for extracting knowledge is to give an idea of the versatility of $\mathcal{FCA}$ as a useful technology for deriving implicit conceptual structures in a plethora of application areas. The next section takes a look at $\mathcal{FCA}$ software tools with the objective to find and justify which tool we eventually chose for our needs, given our constraints and requirements.

A quick review of some available $\mathcal{FCA}$ tools

There exist several software tools designed to exploit $\mathcal{FCA}$ in several ways. They differ by the various features each of them possesses. As we proceeded to understand which system had which

features, we realized that we were finding ourselves in a situation where $\mathcal{FCA}$ was a method we could use for the very purpose of classifying the set of $\mathcal{FCA}$ software systems we analyzed. So it came to reason to use $\mathcal{FCA}$ to help us make a rational choice for the eventual system we chose, having features as per our requirements. And so did we proceed.

ToscanaJ [7][14] is an $\mathcal{FCA}$ software tool developed in Java that allows to manipulate binary and multi-valued contexts, as well as mixed and even nested. The specificity of ToscanaJ is that it enables components to be plugged in though standard interfaces. In particular, this enables it to connect with any DB engine capable of processing SQL queries.

Lattice Miner[15] is another $\mathcal{FCA}$ software engine, also developed in Java at *Université du Québec en Outaouais*, in Canada. It is open-source. It allows all types of contexts and is capable of displaying a graphical rendition of the inferred lattice. It is also able to derive association rules among object attributes from raw data. The user can then review the derived rules to be confirmed with the level of confidence wished for such rules.

There are other Java-based software systems related to Lattice Miner, such as Yevtushenko's Concept Explorer (ConExp),[16] or Valtchev's Galicia [29].[17] Both ConExp and Galicia extend Lattice Miner's capabilities by offering a choice of multiple $\mathcal{FCA}$ algorithms (for lattice construction, lattice drawing, and visualization of dependencies).

Google's OpenFCA (Confexplorer) is an online tool.[18] Its $\mathcal{FCA}$-processing module for computing concepts and links between them on large datasets is called Conexplore. It was written in C# and interacts with two other modules written in Flex:[19] Conflexplore—for context editing, concept computation and visualization, implication computation and attribute exploration—and (a modification of) SpringGraph,[20]—for displaying a set of items that are linked to each other. Flex was originally developed by Adobe, which donated it to Apache in 2012. It is an open-source application framework for building mobile applications for various operatng systems run by portable devices (iOS, Android, BlackBerry Tablet), as well as traditional browser and desktop applications using the same programming model, tool, and codebase. The principal added benefit this gives OpenFCA is that one can use $\mathcal{FCA}$ online on networked resources directly; *e.g.*, through a browser.

Besides full $\mathcal{FCA}$ systems, there are also $\mathcal{FCA}$ tools offering basic $\mathcal{FCA}$ services. For example, Stone,[21] that was developed in Perl to allow converting data files among diverse $\mathcal{FCA}$ formats, or In-Close[22] that is an interactive command-line $\mathcal{FCA}$ console.

While reviewing the above set of $\mathcal{FCA}$ software systems and experimenting with each to obtain some basic user familiarity, we could thus proceed to elaborate a formal assessement of these tools according to their features, insofar as our interests were concerned for the objective of our project. Of course, using $\mathcal{FCA}$ for this purpose made sense. This was invaluable both as an exercise in getting a feel of the method, as well as providing us for a rational systematic analysis of which

[14] http://toscanaj.sourceforge.net/
[15] http://lattice-miner.sourceforge.net/
[16] http://conexp.sourceforge.net/
[17] http://galicia.sourceforge.net/
[18] https://code.google.com/p/openfca/
[19] http://flex.apache.org/
[20] http://mark-shepherd.com/blog/springgraph-flex-component/
[21] http://fcastone.sourceforge.net/
[22] http://inclose.sourceforge.net/

tool would be most appropriate for our objectives under our pragmatic constraints.

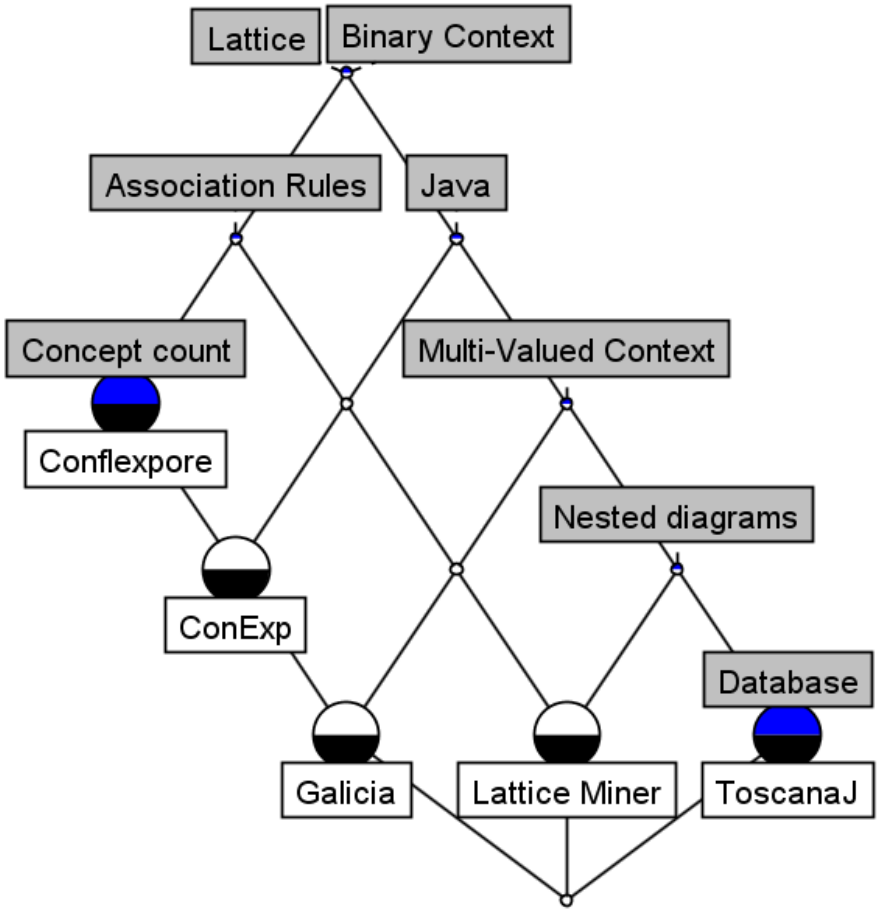

Figure 6: $\mathcal{FCA}$-derived classification of the $\mathcal{FCA}$ software tools we reviewed

Figure 6 shows the taxonomic classification of the $\mathcal{FCA}$ software tools we reviewed resulting using $\mathcal{FCA}$. What we considered in our analysis were the tools as objects with their most salient attributes such as support for binary and/or multi-valued contexts, graphic rendering of derived lattices, management of nested lattices, computation of dependency rules, database connectivity, whether the software is open-source, what source language, *etc.*, ...

Recall that our objective consisted in being able to derive taxonomies and terminological schemas from raw data for the purpose of using them for ontological reasoning and querying. We looked both at deterministic and non-deterministic approaches in Machine Learning. We focused on $\mathcal{FCA}$ as the most appropriate for our needs under our constraints among the other several other relevant technologies that we reviewed [10]. The criteria that guided us in making a choice of technique is that $\mathcal{FCA}$ allows inferring a taxonomy from attributed data, provides efficient practical algorithms to compute and draw concept lattices, and enables grouping attributes and deriving dependency rules among attributes. Such algorithms are both simple and powerful, and easy to adapt to our

needs. In short, we found $\mathcal{FCA}$ to be useful, easy to use, and simple to modify for what we had in mnd to do.

After experimenting with and classifying several $\mathcal{FCA}$ tools as explained above, we set our final choice on Lattice Miner,[23] as it is open-source, has well-documented code, implements an efficient $\mathcal{FCA}$ engine and graphical rendering of concept lattices. Here too, our pressing needs and binding constraints dictated this choice. Rather than sophisticated functionalities such as multiple algorithms and clever lattice rendering and editing, ease of use and being able to get quickly familiar with source code for modifiability were paramount criteria for us.[24]

4.4 How we used $\mathcal{FCA}$ in LivEMUSIC

Software architecture

Task T2's contribution to the project provides a basic module using $\mathcal{FCA}$ for deriving attributed taxonomies from raw data according to given levels of approximation on these data as specified by the user in the form of operationally enforceable constraints. As illustrated in Figure 1 in Section 2, it hinges between Task T1 responsible for identifying a usable dataset from the Data Grand Lyon platform and Task T3 responsible for putting all the parts of the projects together, including taxonomic reasoning. It derives an attributed taxonomy from raw data to be used as a conceptual ontology in Task T3 by the $\mathcal{CEDAR}$ TBox reasoning engine [3, 6, 1]. The $\mathcal{CEDAR}$ system puts the derived attributed taxonomy in consistent normal form (or detects inconsistency). On such a normalized consistent taxonomy, $\mathcal{CEDAR}$ can then process a conceptual query in the context of the derived ontology. It does so by (1) putting the query in consistent normalized form *w.r.t.* the ontology (or found inconsistent, as the case may be), (2) then compiling it into (optimized) $\mathcal{SPARQL}$ form which, (3) when executed on an ABox RDF dataset by a $\mathcal{SPARQL}$ engine, returns responses corresponding to the desired level of semantic approximation expressed by the user.[25]

In designing Task T2's experimental software for what it needed to achieve, we made the following assumptions:

- data are not to be limited to any particular domain: it could come from varied sources (databases, sensors, *etc.*., . . .);
- data are in some normal form legible by the $\mathcal{FCA}$ tools they are to be processed by.

Task T2's inner architecture is depicted in Figure 7. Thus, Task T2's process begins with reading data off a text file (JSON). Using a GUI tool we implemented, the user may configure the taxonomy by choosing relevant attributes from the formal context. The user can also specify the sort

[23] http://lattice-miner.sourceforge.net/

[24] In opting for Lattice Miner, we essentially took into account the strict time constraints imposed on us by Sofiene Chamakhi's academic requirements within the LivEMUSIC project. It was also important that he be the one making this choice to maximize confidence in providing what was expected of him. This being acknowledged, it was yet necessary to implement our contribution (generalized input format and semantic zooming) to support our use scenario of the Velo'V dataset so all of it work with Lattice Miner, which we did with relative ease thanks to the clarity of its well-commented source code. We were fortunate in that regard since there seems to be no other external documentation of this system that we could find anywhere. However, the software is easy to install and simple to use.

[25] See Section 5.

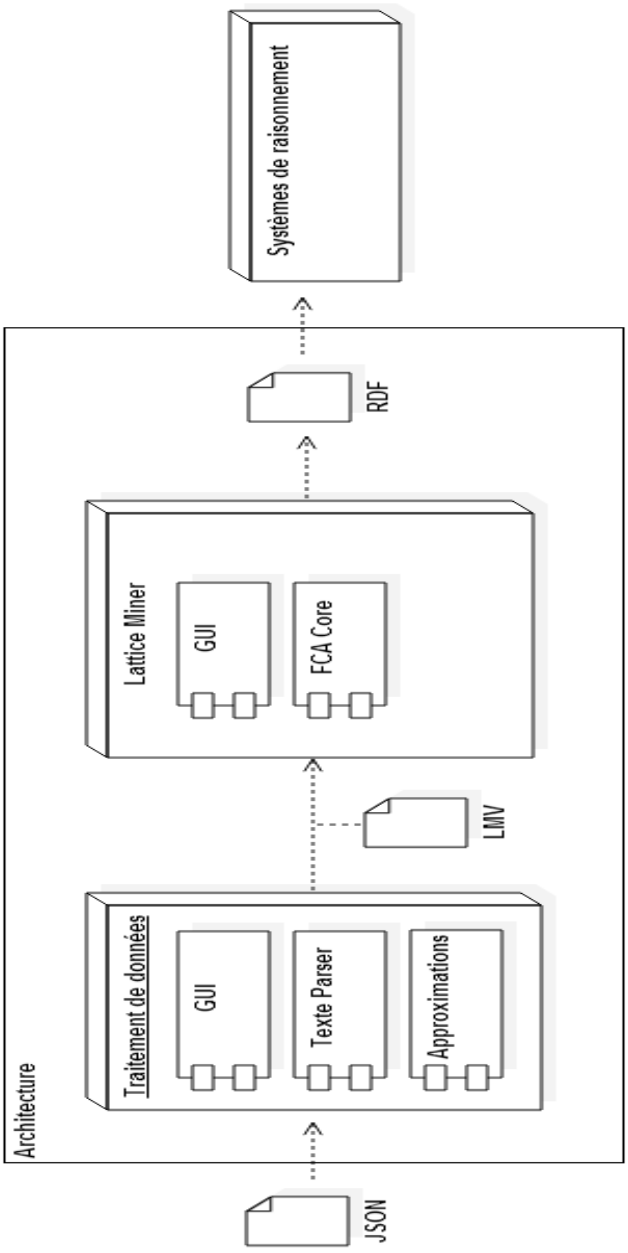

Figure 7: Task T2's inner architecture

of approximations s/he wishes to make on these attributes, thus controling the number of approximated values. Once this is done, the user may activate Lattice Miner, the $\mathcal{FCA}$-based data mining tool, to generate an initial corresponding multi-valued formal concept context. The multi-valued domains of the attibutes we consider are discrete finite sets of values (or value approximations, each standing for a set of approximated values). If no modification to this formal context is performed (*e.g.*, by varying the approximation level on some domains), a binary context is obtained by mapping each multiple-valued domain into as many pairs of attributes as there are values in each domain (one for "yes" and one for "no"). Such generally provokes an explosion; hence the need for means to control the size using approximations. From such a binary context, a lattice is then produced and displayed in graphic form. Finally, the lattice is converted into RDF format to allow using by reasoning tools.

We next summarize each stage of this process.

Data input Lattice Miner's data input format is text of a specific format (LMVT). On the other hand, data from the Grand Lyon has its own format (JSON).[26] Hence, the first thing to do was to modify Lattice Miner's input processor to adapt it to ingest data in the JSON format used for representing data records.

Data customization As discussed in Section 3, the data made available to us on the "Data Grand Lyon" platform are classified according to themes such as transportation (state of the traffic in real time, cycling network management, parking areas, subway stations, *etc.*, ...), healthcare, environment, *etc.*, ... Each source has data structures consisting of several attributes, all of which are not necessarily relevant or exploitable for a specific use. In addition, one must keep in mind that too many attributes would entail a combinatory explosion for $\mathcal{FCA}$ (especially for binary contexts) as it would lead to an enormous number of generated concepts. We addressed this issue by offering the user an interactive service prior to feeding actual input to Lattice Miner that enables customizing which specific attributes to retain for a given application.

Value grouping For some fields (*e.g.*, those with numerical types), it is possible to find several possible (multi-valued) attribute values to be "close enough" for a given purpose. If, in the final binary conceptual context, we were to generate an attribute for each possible value, this would blow up the number of attributes, and therefore of concepts as well. To palliate this issue, we developed a system whereby values of a given type may be grouped into approximation subsets such as intervals or similarity classes. This enables the user to control interactively the desired approximation level per attribute, thereby reducing considerably the number of eventual attributes as per the use one has in mind. This grouping is done by conceptual scaling and can also be defined for data types other than numerical.

Semantic zooming An essential contribution of our work is a notion of "*semantic zooming*." Due to the large size of conceptual taxonomies arising from considering a large number of values for attributes, it is important to give the user a means to control the visual rendition and navigation in oversized formal conceptual lattices generated for such contexts. While many $\mathcal{FCA}$ systems do

[26] See Fonctionnement des webservices numériques.

provide such means at the display level, this zooming facility is purely a graphical functioality: it can only minimize or magnify the dimensions and level of detail of objects being visualized, without modifying their intrisic characteristics. This is "*structural zooming*" in that it has no notion of what the contents mean. It cannot understand whether a user is more interested in more detail for some attributes and less, or none, for others, nor can it summarize the larger conceptual lattice according to semantic properties verified by the values of some attributes. In order to be able to provide such functionalities, one must go beyond just structure and use filtering capabilities based on meaning. This is this capability that we developed in $\mathcal{L}ivE\mathcal{MUSIC}$. We added and demonstrated a basic semantic zooming capability. We did so by adaptating Lattice Miner with generic features to that end and demonstrated instances of these features for a given use case with the Velo'V dataset. What we implemented, although still limited in terms of generality and power of specifiable constraints, allows the user to specify which attributes are of interest and a few appropriate semantic properties enforceable as constraints verified by the values of these attributes. Of course, we could have generalized this to more sophisticated constraints given more resources (time and manpower). For $\mathcal{L}ivE\mathcal{MUSIC}$, we just focused on demonstrating the principle and its actual usefulness on a specific use case.

For the use case we opted for (see Section 3),[27] we considered semantic zooming with approximations for spatio-temporal data. For example, geographical data such as latitude/longitude, local maps, *etc.*, ..., and temporal data such as time of day, day of week, *etc.*, ... The threshold for grouping can thus be specified from a few meters (streets, neighborhoods, *etc.*, ...) to kilometers (town, communal area, *etc.*, ...) The same for time. Both locations and dates can also be estimated with respect to reference points within a given (spatial or temeporal) distance.

Context conversion Spatio-temporal approximations effectuated by the semantic zooming facility can thus allow a user to find a desired level of detail from raw data and thus control the explosion of attributes interactively when converting a multi-valued formal context to a binary one by grouping all similar values into a single one. In mathematics, this is formalized as defining an equivalence relation on the set of data and replacing the full dataset with its quotient set *w.r.t.* this equivalence relation. The equivalence relation on a dataset consists in considering two individual records similar up to the user-specified constraints. In our demonstrated use case's dataset (Velo'V), only deterministic spatio-temporal constraints are supported. However, any other kind of semantic (*i.e.*, value-based rather than structure-based) constraints, including non-deterministic (*e.g.*, probabilistic, fuzzy) on a given dataset could be used as long as it may be enforced operationally on the data.

Implementation

All the above was realized as an extension of Lattice Miner.

Data input and customization We added a new module to Lattice Miner implementing a GUI tool extending the whole system it with a JSON reader and an LMVT writer to be used upstream of the other Lattice Miner modules. This GUI tool was done using the standard Java Swing

[27] A use scenario making helpful intelligent use of Velo'V data.

`javax.swing` toolkit libraries.[28] This tool eases the processing of JSON input as a record-field and interaction with the user for attribute selection to generate from it an object-attribute formal contexts. It parses data in JSON format and displays all available fields to the user who can then interactively pick and choose which objects and attributes are relevant (see Figure 8).

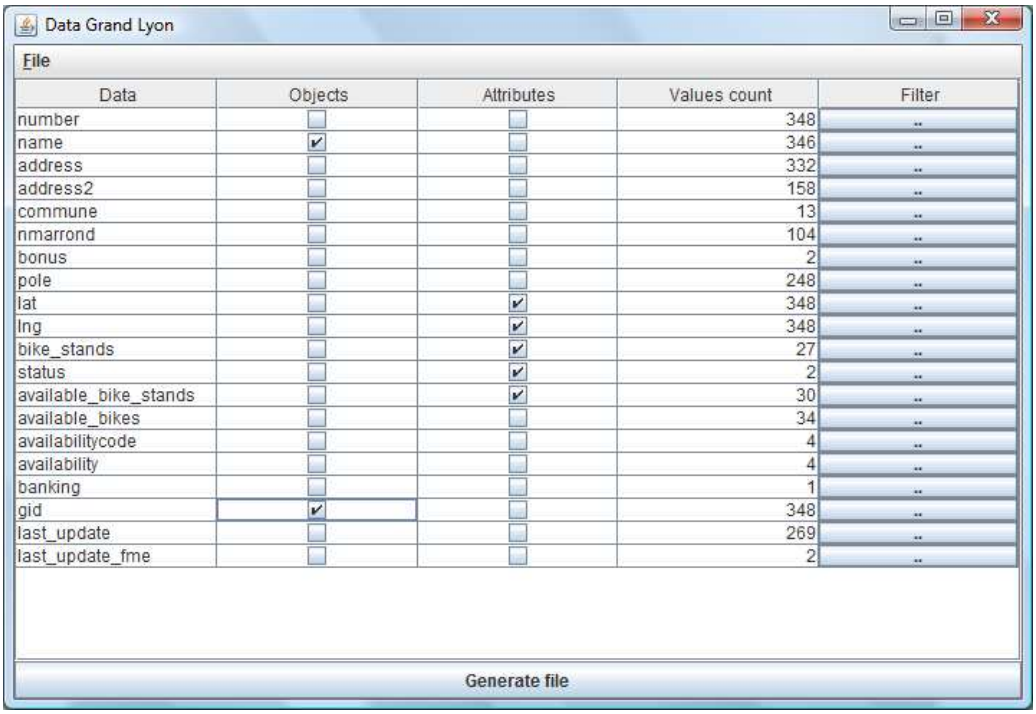

Figure 8: JSON-reading LMVT-generating GUI front-end for object and attribute selection from available data

We took care to implement this tool keeping reusability in mind. In particular, it is a simple matter to plug in text-data input modules other than than one for the JSON format (*e.g.*, XML, CSV, RDF . . .), or to modify the way the graphical interface renders a conceptual context imply by changing a few global parameters. The GUI rendition of the raw dataset we implemented for generating a formal context (see Figure 8) displays a 2-dimensional array each row with text field names from the raw data and the number of available values for each field. The user can then pick and choose which of the raw data fields will be interpreted as objects and which will be interpreted as attributes by ticking on the appropriate cell.

Semantic zoom approximations The most innovative part of our implementation concerns offering the user the possibility to augment or diminish the level of precision of attributes among legal values allowed for it. The functionality we developed in the specific use scenario using the Velo'V datasets focused on spatio-temporal zooming. In effect, this realized a form of user-

[28] http://docs.oracle.com/javase/6/docs/api/javax/swing/package-summary.html

controlled conceptual scaling. This is specified through the graphical tool mentioned previously (see Figure 8). This GUI provides a button for each field of the JSON data (rightmost column) that allows the user to specify the desired approximation level.

In our case, we set up three sets of approximations:

- **Geographical data** Among many data accessible through the Grand Lyon's platform, one can find several fields meant to represent geographical coordinates (*e.g.*, a Velo'V station, a TCL bus station, *etc.*, ...). The format for such data is a pair of numerical values standing for latitude and longitude.

  For such geographical data, we provide the following possible conceptual scaling to be used for semantic zooming.

  - *Approximation by nested intervals* The user can specify a set of reference coordinates together with a distance (or a walking time) around it. This is then used to regroup locations into sets of concentric nested intervals of equal width around the reference location.

    Let us take for example someone wishing to use this type of approximation on the Velo'V dataset. S/He could set the reference point to the geographical coordinates corresponding to Lyon's "*Hôtel de Ville*" together with an interval width of 1 km. Then, all the Velo'V stations located at a distance less than 1 km of the *Hôtel de Ville* will thus be regrouped into a single conceptual entity, then those between 1 km and 2 kms from the *Hôtel de Ville* will be regrouped into another conceptual entity, then between 2 kms and 3 kms, and so on.
  - *Approximation by tolerance radius* The user specifies the coordinates of a reference and a distance. These represent repsectiveley the center and radius of a circle within which all coordinates will be considered similar to the center for purpose of conceptual scaling. For example, choosing the center of the La Doua campus of the *Université Claude Bernard Lyon 1* and a radius of 1 km will result in considering all points within this circle equal to the center. An example of the two foregoing approximations is illustrated in Figure 9. In order to apply a geographical approximation in a JSON data, the user must specify the field that corresponds to latitude, and the one corresponding to longitude, as well as the type of approximation along with either the interval width or the radius, depending on which of the two kinds of geographical approximation is desired.

- **Temporal data** We provided another similar approximation for temporal data. Namely, one that selects a moment in time in the form of a date and time of day, along with a time before and after, thereby allowing the user to define a lapse of time to be considered as the same point in time.

- **Numerical data** Approximations on numerical data are similar to geographical data. Namely, it is possible to regroup the values of numerical attributes according to nested equal-width intervals around a value:

$$\bigcup_{k\geq 0} \big(\,]v-(k+1)r, v-kr] \;\cup\; [v+kr, v+(k+1)r[\,\big)$$

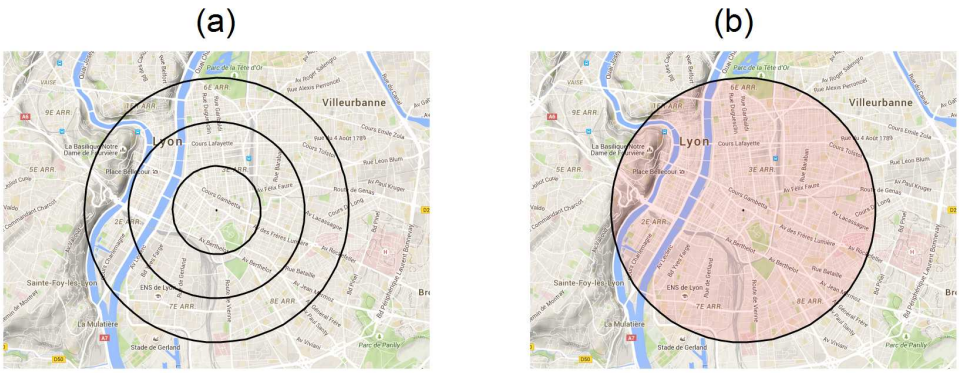

Figure 9: Geographical approximation by (a) nested intervals or (b) tolerance radius

or to whether they belong or not to an interval $[v - r, v + r]$ centered in a value v with a specified tolerance radius r.

In our experimental implementation, we assume a uniform distribution of the data in such intervals (see Figure 10). However, as we discuss in Section 7.3, it would also be possible to consider other data distributions.

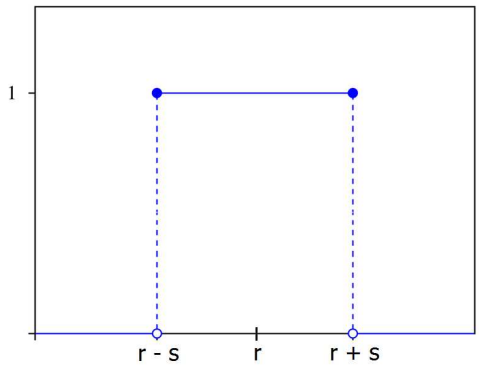

Figure 10: Uniform distribution over grouped data

In the GUI we implemented, it is possible to make approximations on attributes thanks to a button dedicated to each field. When clicking on the button for a given attibute, the user can thus select and specify the parameters for the desired approxmation for this field. This is illustrated in Figure 11 where an approximation of type "tolerance radius" is being specified.

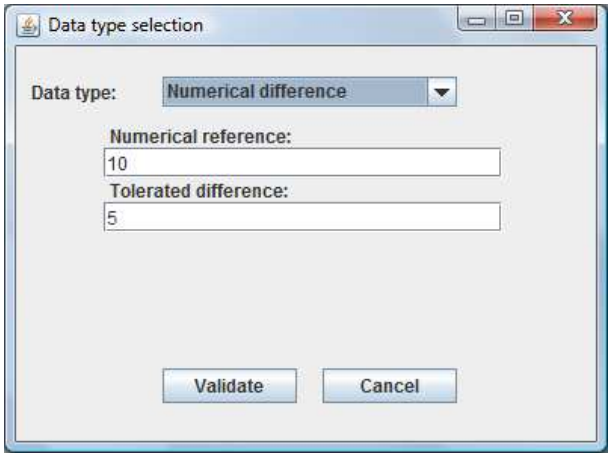

Figure 11: Specifying a numerical value approximation

Experiments

The point of the $\mathcal{L}ivE\mathcal{MUSIC}$ project being to experiment on the real data provided by the Data Grand Lyon platform, and verify that our approach is applicable on a real use case. This hinges on deriving from raw data ontologies that can be customized by the user in a semantic way.

The use case we focused on was one exploiting the Velo'V dataset keeping temporal data on the states of all the Velo'V stations in the Grand Lyon region. This dataset contains an entry for each station (348 in total). Each entry per station contains 20 data fields consigning station's state of use. Among the fields, there are data encoded as character strings (*e.g.*, name of the station, its address, *etc.*, ...), numerical data specified as integers (*e.g.*, station's total capacity, number of currently available bikes, *etc.*, ...), boolean data (*e.g.*, payment by card accepted, *etc.*, ...), temporal data (*e.g.*, date, time of day, *etc.*, ...), geographical data (*e.g.*, GPS location,*etc.*, ...).

We experimented with the test dataset shown in Figure 8. The conceptual context contained 348 objects (Velo'V stations) and we selected 4 attributes (latitude/longitude form a single attribute). We proceeded with the following sets of test:

- **Case 1**: derivation of a binary $\mathcal{FCA}$ context without any approximation; approximation.
- **Case 2**: making one approximation on geographical data by nested intervals of width equal to 1000 m;
- **Case 3**: making one approximation on geographical data by nested intervals of width equal to 2500 m;
- **Case 4**: making one approximation on geographical data by nested intervals of 2500 m width, one approximation on the total number of available bikes by packets of 20, and one approximation on the number of available bikes bewteen 10 and 20.

As shown in Table 1, we could verify that, indeed, the rougher the approximations, the less explosive the size of the generated concept lattice.

Test case	# of concepts
Case 1	418
Case 2	266
Case 3	181
Case 4	44

Table 1: Number of concepts generated per test case

This technique is efficient in that it enables considerable downsizing the generated concept lattice, thereby sparing the user having to extricate useful information from oversized concept lattices. The advantage is that, unlike structural zooming displays, this grouping is semantic: it is performed in a coherent manner according to conceptual proximity metrics that are defined by the user using abstract properties of relevant data.

5 Task T3: Use of Ontological Knowledge for Monitoring Applications

This task was assigned to Mathieu Veyrand under Hassan Aït-Kaci's supervision (Mid-June–October 2015).

5.1 General view

As depicted in Figure 1, the LivEMUSIC project was decomposed into four successive tasks, the first three of which were:

- Task T1: Selecting an cleaning up datasets from Data Grand Lyon (DGL);
- Task T2: Extracting an attributed ontology from raw data with Lattice Miner;
- Task T3: Ontological reasoning with $\mathcal{CEDAR}$ and $\mathcal{SPARQL}$ query processing.

While Task T1 concerned dataset selection and cleanup from those made available by the Grand Lyon and Task T2 extracted an ontology appropriate to the user from this raw data, Task T3's objective was to put all the pieces together. The challenge was to ensure that Task T2's software tools could be effectively conjugated with the $\mathcal{CEDAR}$ reasoner and a $\mathcal{SPARQL}$ query processor so as to provide a working prototype demonstrating a complete convincing use scenario using the Velo'V dataset selected from the Data Grand Lyon. This general view of the parts to "piece up" and how they are expected to connect can be summarized as the block diagram shown in Figure 12.

The basic nature of Task T3, and therefore its essential contribution, was to consist more in an experimental software engineering task than an innovative research task. Thus, we will now give an engineer's view of the job that needed to be done, how it was realized, and what pragmatic design decisions it called for.

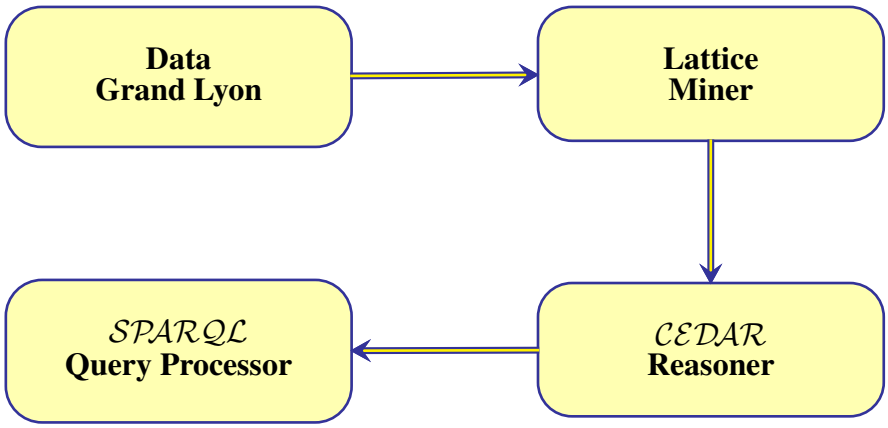

Figure 12: General view of the *LivEMUSIC* processing diagram

5.2 Task T3 as the synthesis of the *LivEMUSIC* project

In terms of running software, the *LivEMUSIC* project was to rely on four parts. What these are and the way they articulate with one another can be summarized as follows.

1. The data interface for the Data Grand Lyon datasets programmed in Task T2 by Sofiene Chamakhi (see Section 4.4). This module, added to the Lattice Miner system, takes data from the datasets that were selected in Task T1 by Tanguy Raynaud (see Section 3.2). These data are input in JSON format. The module then offers a GUI for the user to choose the desired parts of the data upon which s/he wishes to perform semantic approximations. The approximated data are then output in LMVT form (Lattice Miner's own specific format). For the sake of enabling further processing and providing a more general output format, Sofiene's software was adapted by Mathieu Veyrand for Task T3 by defining and adding an RDF output format as well.

2. The Lattice Miner v1.4 system itself,[29] for converting data in LMVT form into a binary lattice, to be then exported in the *OSF* format used by the *CEDAR* reasoner. In addition, for later interpretation intelligible to the user, it was also necessary to generate the information associating the original sets of approximated data (in our use case, the Velo's stations) with the generated concepts approximating them. (See for example Figure 18.)

3. The *CEDAR* v2 *beta* system,[30] for performing ontological reasoning on the attributed-concept lattice generated by Lattice Miner. This system also generates *SPARQL* queries over RDF data needed to retrieve which actual Velo'V stations match the user's desired criteria. It was developed by Hassan Aït-Kaci and Samir Amir [3, 6, 1].

[29] http://lattice-miner.sourceforge.net/
[30] http://cedar.liris.cnrs.fr/cedar-software/cedar/

4. A basic GUI for $\mathcal{SPARQL}$ to process queries and interpret the results into a form legible for the user (*e.g.*, a list of Velo'V stations), which was developed by Mathieu Veyrand.

Implementation and software development

As far as implementation is concerned, each of the four parts above was realized as an Eclipse project. Except for $\mathcal{CEDAR}$, which required importing the required libraries manually, all the needed packages are imported automatically thanks to the use of Apache Maven.[31] We next describe the salient points of each the four parts.

1. **Pre-processing of Data Grand Lyon** This part of the software can be divided into three subparts.

   (a) *Approximation management* This is done by the packages called `approximations` and `approximations.datatype`. The former defines classes for each type of approximation, while the latter defines classes for the corresponding datatypes (numeric, geographic, or temporal). The class `DataType` is the base class for all approximations. Then, there is a class for each type of approximation, each having specific attribute denoting the approximation's components (*e.g.*, for the `Interval` approximation class, attributes for the reference point (lattitude and longitude) and the width between successive levels. All static methods needed for the computation of approximations are centralized in the class `Approximations`.

   (b) *Reading and converting input files* The class `ContextGenerator` manages this part. It consists in methods for parsing JSON-formatted data and convert them into the LMVT format used by Lattice Miner.

   (c) *The user interface* This part is operated through the class `GUI`. The structures needed for displaying the data are defined in the class `MyTableModel`.

2. **Formal concept analysis** This was carried out using Lattice Miner extended with the functionalities we needed that were missing.

   The most notable class is in the package `fca.io.lattice.writer.rdf` and called `LatticeWriter`. It defines the method `writeOSF` used for converting a binary concept lattice produced by Lattice Miner into corresponding $\mathcal{OSF}$ format to be processable by $\mathcal{CEDAR}$. This method is automatically invoked through the class `ContextViewer` as a side effect whenever the user clicks on the button labelled "**Show Lattice**" in the GUI.

   The $\mathcal{OSF}$ format is text-based and describes a concept taxonomy comprising $\mathcal{FCA}$ generated concepts and additional concepts for the values of the generated concepts' attributes. (See for example Figure 20.)

   It is quite simple; it starts with a line of the form:

   ```
   Concept_0, int, ... <| Thing.
   ```

[31] https://maven.apache.org/

where "`Thing`" is the topmost overall concept, "`Concept_0`" is the maximal $\mathcal{FCA}$-generated concept, followed by other identifiers such as "`int`," *etc.*, ..., corresponding to any additional needed attribute-value concepts. The symbol "`<|`" means "*is a*" and denotes the subconcept ordering of the taxonomy.

Then, this is followed by lines of the form:

```
Concept_1, Concept_2, ... Concept_n <| Concept.
```

for a concept "`Concept`" and its immediate children concepts.

The OSF-format specification of attributes is of the form:

```
attribute_name << Concept.
attribute_name >> AttributeConcept
```

to specify that an attribute named "`attribute_name`" associates to a concept called "`Concept`" the concept called "`AttributeConcept`."

3. **Ontological reasoning** This was carried out using the $\mathcal{CEDAR}$ V2 beta system.[32]

   It was only necessary to modify $\mathcal{CEDAR}$'s code in a few minor points in its GUI. The action activated by executing an $\mathcal{OSF}$ query was altered. When the option labelled "**retrieve instance**" has been checked (see Figure 21), the actual instances making up the resulting concepts (*e.g.*, Velo'V stations) are retrieved from the concept/instance file that was passed from the initial approximation module and displayed in the GUI (*i.e.*, one such as shown in Figure 18). Also, clicking on the button "**Export results**" now launches a $\mathcal{SPARQL}$ client to carry out the query processing.

4. **RDF data querying** This was performed in $\mathcal{SPARQL}$ executing the code generated by $\mathcal{CEDAR}$ from an $\mathcal{OSF}$-normalized conceptual expression it verified to be semantically consistent.

   This part was realized as a basic GUI used to process $\mathcal{SPARQL}$ queries, together with a filter mapping object ids (*e.g.*, internal reprsentations of Velo'V stations) to user-legible names. It was generated as an Eclipse custom window. The processing of the query uses ARQ, the Apache Jena $\mathcal{SPARQL}$ library.[33]

   A typical $\mathcal{SPARQL}$ query generated in our Velo'V use case is of the form:

```
PREFIX ex: <http://data.grandlyon.com/>
SELECT ?y ?id
   WHERE {
      ?x ex:Address ?y ;
      ex:Id ?id.
FILTER (?id="1025" || ?id="1542" ... ) }
```

   to return the addresses of all the stations corresponding to a user's criteria at a specified approximation zooming.

[32] http://cedar.liris.cnrs.fr/cedar-software/cedar/
[33] https://jena.apache.org/documentation/query/

In Appendix Section B, we make additional points concerning a few pragmatic design choices we had to make for Task T3 to combine the parts comprising it.

6 Task T4: Project documentation and software demonstration

In the course of the $\mathcal{L}ivE\mathcal{MUSIC}$ project, we took care of reporting and documenting our work in various ways. All technical documents and software documentation we produced are accessible online.[34] This task consisted in ensuring that all participants' work be repertoried as due and that all this information be set up properly for easy access. It was the responsibility of the project's director, Prof. Hassan Aït-Kaci.

We itemize this information below.

Project overview and results report

- $\mathcal{L}ivE\mathcal{MUSIC}$ Final Report (this document)—[pdf][35]

Auxiliary documents per task

- Task T1 Preselected Datasets from the Data Grand Lyon—[xlsx][36]
- Task T2 Sofiene Chamakhi's MSc report (in French)—[pdf][37]
- Task T3 Overview—[pptx][38]

Software demonstration

- Demo Video—[YouTube][39]
- Instructions to download the software and run the demo—[html][40]

Background documents

- Original LivEMUSIC Proposal—[pdf][41]
- Revised Proposal—[pdf][42]

[34] http://cedar.liris.cnrs.fr/livemusic/
[35] http://cedar.liris.cnrs.fr/livemusic/doc/livemusic-final.pdf
[36] http://cedar.liris.cnrs.fr/livemusic/doc/Preselection-Data-Grand-Lyon.xlsx
[37] http://cedar.liris.cnrs.fr/livemusic/doc/CHAMAKHI_Sofiene_2014-2015_TIWE.pdf
[38] http://cedar.liris.cnrs.fr/livemusic/doc/LivEMUSIC-TaskT3-overview.pptx
[39] https://www.youtube.com/embed/nM3dDy74qIw
[40] http://cedar.liris.cnrs.fr/livemusic/READ-ME.html
[41] http://cedar.liris.cnrs.fr/livemusic/doc/LivEMUSIC-proposal-orig.pdf
[42] http://cedar.liris.cnrs.fr/livemusic/doc/LivEMUSIC-proposal-reduced.pdf

7 General Discussion

All parties of the $\mathcal{L}ivE\mathcal{MUSIC}$ project (Région Rhône-Alpes, Université de Lyon, Université Claude Bernard Lyon 1) are aware that this project is a reduced adaptation of the original proposal as submitted.[43] The reduction in funds led to a revised proposal with more modest objectives.[44] This revision had to be adapted further for lack of suitably qualified student candidates, replacing two planned Master-2 projects with part-time engineering assistance within the granted funds as explained below.

7.1 Roadblocks

Because of the revisions of the original proposal imposed by the actual granted funds, it was necessary to modify the structure and schedule of $\mathcal{L}ivE\mathcal{MUSIC}$, as described in Section 2.[45] However, the new more pressing time constraints (one year instead of two expected) made it difficult to find and recruit three suitable Master-2 candidates for the three tasks T1–T3. From the initial postings in November 2014 and several repostings as late as February 2015, on diverse national bulletin boards such as Bull-i3, we reviewed over 10 applications. We conducted full interviews for at least 6 candidates. Two offers that we made were rejected at the last minute before the project actually started (end of January 2015). Despite our best efforts and several urgent repostings, we could see no tenable Master project candidates except for one (Sofiene Chamakhi for Task T2). As a result, we asked for permission to redirect funding that had originally initially been budgeted for two Master-2 full-time internships toward support of only one part-time Assistant Engineer.[46]

We obtained the requested permission, and hired Tanguy Raynaud who had obtained his Master-2 in 2014 in the course of the $\mathcal{CEDAR}$ Project as our part-time engineer from end of February until end of September 2015. However, he resigned at the end of April upon being offered a PhD scholarship. We could find a suitable candidate to replace him under the same conditions : Mathieu Veyrand, who had obtained his Master-2 in Computer Science at the LIP (Laboratoire de l'Informatique du Parallélisme) in 2014.

Due to the rather stringent constraints explained above (essentially lack of available personnel and time), the project focused on pragmatics, while still maintaining a challenging objective.

7.2 Achievements

The work presented in this document assembles techniques from diverse cutting-edge AI fields for exploiting raw data into a demonstrable proof of concept (software demo). This project shows how combining tools capable of (1) learning from data, and (2) reasoning with the concepts learned, may provide useful applications exploiting available data regularly gathered from sensors monitoring a plethora of activities in an urban milieu such as made available by the Data Grand Lyon platform. In that, it demonstrates how to make "intelligent" use of the kind of data continuously

[43] http://cedar.liris.cnrs.fr/palse/doc/LivEMUSIC-proposal-orig.pdf
[44] We had originally requested 441 273 € for a two-year project; we were granted 150 000 € for a one-year project.
[45] http://cedar.liris.cnrs.fr/palse/doc/LivEMUSIC-proposal-reduced.pdf
[46] *Ingénieur d'études à temps partiel.*

being recorded from urban life activities.

A crucial aspect of what we had set out to achieve was that we were to work from raw data (as opposed to well formatted and organized databases), making things much more challenging than otherwise thought. Indeed, such data are gathered with no specific intended use and are thus in raw form using primitive types (number, strings, *etc.*) and no logical conceptual typing. Fortunately, we demonstrated that data-mining techniques such as $\mathcal{FCA}$ could infer conceptual structures from it. We controlled the size explosion of the derived conceptual lattices using operationally enforceable constraints on structured data verifying abstract semantic conditions. This enables a user to make approximations at the proper level of details, thus adjusting the focus on meaningful taxonomies to prevent having to deal with uselessly over-detailed information. We dubbed this "semantic" zooming (as opposed to merely "structural" used in GUIs for ontology displays). These semantic approximations may then constitute the intensional schema of an attributed ontology (Terminological Box) than can be verified consistent and classified using an ontological reasoning system. The latter can also be used to generate from conceptual queries optimized data queries on the extensional data associated to its conceptual structures (Assertional Box). This is akin to what is known as "Abstract Interpretation" in programming-language verification [13, 14], with the added freedom to specify one's desired level of abstraction as fits one's needs. The simpler abstraction logic of attributed ontologies (compared to that of more general abstraction logics needed to reason about general-purpose programming languages) was conducive to making this feasible in this particular setting.

7.3 Potential extensions

This prototype can of course be improved in several ways. The two most immediate two potential extensions we could envisage are (1) in the derivation of more sophisticated ontological knowledge from raw data, and (2) dealing with uncertainty. A third dimension, with even greater potential, will be (3) to go beyond mere conceptual reasoning and data querying by also providing services both for descriptive and prescriptive Analytics such as On-Line Analytical Processes (OLAP).[47]

Ontological desiderata

The original perspective we have experimented with in this project has been to try and derive ontological structures from raw data by, in effect, mining/learning concepts by clustering these data into semantic classes specified using constraints to be verified by the data comprising their extensions. The gist of this process is akin to abstracting a set of elements (the data) into a *quotient* set of equivalence classes of these elements (the concepts).

What we demonstrated was necessarily limited in the kind of constraints to experiment with, as well as focused only for use scerarios dealing with the specific datasets on which we demonstrated the viability of our approach (*viz.*, Velo'V). In general, this ought to be generalized to larger classes of constraints that could be enforced on varied datasets. In fact, it would be beneficial to categorize such constraints and associated methods for enforcing them to span as large a toolkit as possible. Richer ontologies (not just attributed), would then arise which more powerful reasoners could leverage in the way we illustrated (albeit on basic conceptual taxonomies).

[47] https://en.wikipedia.org/wiki/Online_analytical_processing

Uncertain data

Another extension's dimension that begs experimenting is to allow more expressive constraints that just position and time differences or intervals. Being able to handle some uncertainty in the nature of the data and still manage to provide coherent adapted assistance.

There can be several formal notions that may be made operational for reasoning under uncertainty; to cite the most popular and the most used: Bayesian probability [22] (*e.g.*, with condition/action rules [4]), Fuzzy "if/then" rules [17], "Soft" constraints [8], Rough Set estimates [23], *etc.*, ...

For example, as we mentioned in Section 4.4, it would also be possible to consider other data distributions (*e.g.*, gaussian, exponential, *etc.* ...). Thus, instead of a strict uniform distribution of importance weights such as used in Figure 10, we could consider normal (gaussian) distributions (see Figure 13). For example, this would be used to modulate the importance of the data in more nuanced expressivity (*e.g.*, the further Velo'V stations within a 1000 meters radius from my current position are of lesser preference than the closer ones).

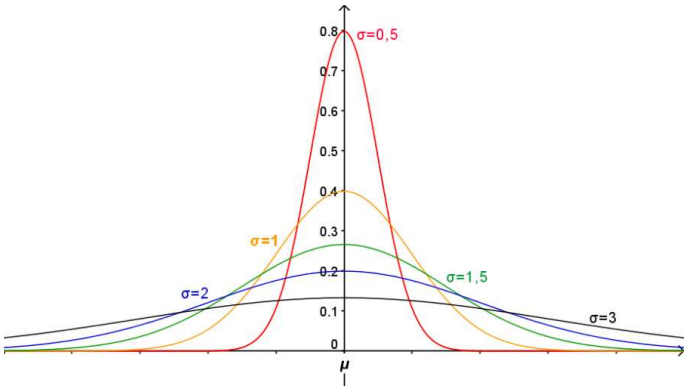

Figure 13: Gaussian distributions over grouped data

Analytics

This section summarizes the essential contents of Hassan Aït-Kaci's invited presentation at the 2015 Workshop on Constraint Programming and Analytics, as a member of a panel of experts [2].[48]

Computer-aided reckoning has shifted from mere data processing, as clever as it has been, to deriving, and using, knowldedge from massive data. The latter point, exploiting the fact that data have indeed become large enough to make statistical inference perform on a par with, and indeed as a nice complement to, any other "intelligent" control (whether predictive or prescriptive).

As a result, the computing world is currently undergoing a profound metamorphosis. Developing decision-making application software is no longer about how to devise clever constraint-solving algorithms. To a large extent, there already exist a plethora of constraint-solving implementa-

[48] http://booleconferences.ucc.ie/cp2015workshops/cpandanalytics

tions capable of handling large and complex constrained optimization models. Many successful such algorithms have been packaged into well-honed implementations that meet industry-strength requirements (*e.g.*, IBM's ILOG Solver,[49] Google [24]). Rather, the difficulty now being faced regards the specification, orchestration, deployment, and maintenance of how to organize such constraint-solving modules for the purpose of specific applications. Such applications are no longer stand-alone optimization problems, but dynamic combinations of Analytics, Optimization, and Decision-Making under uncertainty (hence reliance on statistical tools).

Therefore, the hardship in putting together such multi-tasked applications is not in the implementation of constraint-solving algorithms for various classes of constraints being handled (these are abstracted away as interfaceable constraint-solving library modules). It is in how easily one can organize such a choregraphy of modules, some of which CP-based, for the purpose of applications interacting with other easy to interface units for data analytics, plotting, reporting, and generic numerical data processing. Informally, it is not about the various modules of such a system, but about the "glue" that holds these parts together and make them interact. This glue is the *scripting*.

What is *scripting*? Isn't it programming? Isn't writing a Javascript app the same as writing a Java program? Or is it? Yes and no. On one hand, scripting is indeed a form of software programming in the sense that it is writing an executable coded specification of instructions. On the other hand, it also differs from classical compile-and-deploy static applications. Static applications are well-honed high-performance software modules implementing the best *algorithms* on given data. But,

- scripting programs are not statically compiled then executed; they are dynamically interpreted text-based source code—in particular, they can be put together as strings of text and executed on the fly;
- scripting is specifying how to orchestrate several programs into a coherent whole made of static application modules;
- scripting is "light-weight" programming where the focus is not on application algorithms, but on the clever use of a very large pool of available libraries: public[50,51] and, of course, private.

Hence, scripting is more useful for specific purposes such as web-oriented visual-oriented dynamic jigsaw puzzle construction—indeed more like solving a constraint problem using available building blocks, and being able to build new ones and re-use them as one's own libraries.

The state of the art for scripting data-driven decision-making applications is using existing available scripting idioms (essentially Python and JavaScript). Putting together this sort of scripting glue is far from trivial and constitutes the most arcane part for coordinating constrained optimization and data analytics. The problem, then, is how such scripting could be made less *ad hoc* and more systematic—in particular, easier to set up, understand, and adapt.

[49] http://www.cs.cornell.edu/w8/iisi/ilog/cp11/usrsolver/usrsolverpreface.html
[50] [...]TheBigGlossaryOfOpenSourceJavaScriptAndWebFrameworksWithCoolNames.aspx
[51] http://speckyboy.com/2012/11/26/dynamic-open-source-javascript-libraries/

8 Conclusion

This document has overviewed work that was carried out as part of the *LivEMUSIC* Project (*Living Environmnent Monitoring Use Scenario with Intelligent Control*). This work combined technologies for learning from data, knowledge representation, and ontological reasoning and conceptual querying.

What this project meant to illustrate on an actual dataset (Velo'V dataset) taken from those some 500+ available from the Data Grand Lyon platform is that such a combination is possible. The point to make was a proof of feasibility. There could surely be other, and more sophisticated, ways to proceed given more time and manpower resources.

What was achieved in this project is therefore a clearly favorable argument for endowing a more ambitious research project with sufficient (time and human) resources. Such a project would develop general techniques that could be used to build a truly versatile and powerful system for producing a diversity of intelligent uses exploiting general-purpose raw data gathered by urban sensors and public-application monitors.

Appendix

A Initial Exploitable Datasets Selected in Task T1

The following table lists the 38 datasets from the Data Grand Lyon that were preselected in Task T1. Each dataset follows a primitive relational record schemas consisting of their attribute names and types. Such schemas consist of primitive types (digit, integer, character, character or digit string, *etc.*) or simple types (such as date, URL, *etc.*). For more details on the actual types of records used in of each dataset, what attribute field they have, and in what format they are represented, see this Excel sheet that we produced as the end result of Task T1's analysis effort.[52]

1. *Voies cyclables*
2. *Alertes accessibilité TCL*
3. *Alertes trafic TCL*
4. *Agences commerciales TCL*
5. *Bornes E-tecely*
6. *Bureaux de votes*
7. *Caméra WebCriter*
8. *Chantiers perturbants*
9. *Dénomination des voies*
10. *Emplacement de personnes à mobilité réduite*
11. *Entrées-sorties de métro TCL*
12. *Etat du traffic en temps réel*
13. *Lignes bus TCL*
14. *Lignes metro TCL*
15. *Lignes tram TCL*
16. *Marchés forains du Grand Lyon*
17. *Parcs de stationnement Velo'V*
18. *Parcs relais TCL*
19. *Parkings*
20. *Parkings (temps réel)*
21. *Parkings Criter*
22. *Pluviométrie*
23. *Points d'interêt touristique*
24. *Points d'arrêt TCL*
25. *Points service TCL*
26. *Pôle interêt*
27. *Prochain passage TCL (temps réel)*
28. *Risques allergies pollen*

[52] http://cedar.liris.cnrs.fr/palse/doc/Preselection-Data-Grand-Lyon.xlsx

29. *Relais info-service TCL*
30. *Silos enterrés*
31. *Silos sous verre*
32. *Sites de traitement des déchets*
33. *Stations Autopartage*
34. *Stations Vélo'V*
35. *Stations Vélo'V (temps réel)*
36. Toilettes publiques
37. *Vols temps réel au départ de Lyon Saint-Exupéry*
38. *Vols temps réel à l'arrivée de Lyon Saint-Exupéry*

B Some Task T3 design choices

B.1 Improving data classification with types

Lattice Miner offers no means to define and export data types. It was therefore necessary to devise a way for making data types correspond between Lattice Miner and $\mathcal{CEDAR}$. We set up a simple numeric code scheme to specify the types of attributes using declarations of the form:

```
attribute_name == type
```

based on the four basic types used in Data Grand Lyon datasets. Namely,

```
STRING ("string")
INTEGER ("int")
DATE ("date")
BOOLEAN ("boolean")
```

Then, prior to generating the LMVT file, we attribute static numeric codes per types of attribute. For example, for the Velo'V dataset, we used the following codes:

```
0 : ID
1 : Name (String)
2 : Address (String)
3 : Address2 (String)
4 : Commune (String)
5 : Num Arrondissement (Int)
6 : Bonus (Boolean)
7 : Pole (String)
8-9 : Lattitude / Longitude (DG)
10 : Bike_Stands (Int)
11 : Status (String)
12 : Available_Bike_Stands (Int)
13 : Available_Bike (Int)
14 : Availability_Code (Int)
15 : Availability (String)
16 : Banking (Boolean)
17 : Last_Update (Date)
18 : Last_Update_fme (Date)
```

This type information is transparent to Lattice Miner: it generates the binary concept lattice regardless of it. It is used only when we generate the $\mathcal{OSF}$ ontology.

B.2 Improving the JSON-to-RDF conversion from Task T2 to Task T3

The RDF form generated by the system implemented in Task T2 had no real encoding of the data structures corresponding to the JSON input files providing the actual datasets. It had no real schema, just itemizing all the contents as dumb text [10]. In order to process it further for Task T3, it was necessary to reprogram the JSON-to-RDF generator so as to give it as rich a semantic structure as possible. This was done using standard Semantic Web constructs (see Figure 19 for example). We used the Jena libraries for this purpose: the RDF API,[53] and the Ontology API.[54]

C Use scenario illustration on the Data Grand Lyon Velo'V dataset

This section briefly illustrates a typical use of the coupling of the $\mathcal{CEDAR}$ Reasoner with the enhanced Lattice Miner system now able to generate an attributed ontology from raw data from the Grand Lyon thanks to constraint-based semantic zooming. This can be effectively coupled to a $\mathcal{SPARQL}$ engine to retrieve matching instances. The steps described below are illustrated in a demo viewable online as a YouTube video.[55]

Assume a user wishes to identify all the Velo's stations locate within a 1000 meter radius from where s/he is, and which contains at least 5 available bikes.

- The user first selects a data file and chooses "`number`" as object, then "`lat`," "`lng`," and "`available_bikes`" as attributes. The user then chooses "`latitude`," "`longitude`," and '`numerical threshold`" as filters for each attribute, respectively. Finally, this choice is validated. See Figures 14 and 15.
- The user can then perform the desired semantic approximations through an appropriate GUI as shown, for this use case, in Figure 16.
- Upon validation, the concept lattice can then be generated by Lattice Miner and exported in LMVT format (see Figure 17. In addition, a file associating the ids of actual data instances (*i.e.*, Velo'V stations) to the concept approximating them is passed along (see Figure 18).
- In addition to LMVT format, while still in Lattice Miner, the user can trigger the generation of the same information in RDF form using Semantic Web primitives (see Figure 19). This information is also expressible in $\mathcal{OSF}$ syntax for further processing by the $\mathcal{CEDAR}$ reasoner (see Figure 20).
- Using the $\mathcal{CEDAR}$ V2 *beta* tool (see Figure 21), one can then load the $\mathcal{OSF}$ file to verify the consistency and normalize the desired $\mathcal{OSF}$ query using the classified ontology's Terminological Box—or "TBox" (*i.e.*, the classified conceptual ontology).
- By also checking the "`retrieve instance`" option, a query in $\mathcal{SPARQL}$ corresponding to the normalized $\mathcal{OSF}$ query is generated to be executed on the Assertional Box (or "ABox")—*i.e.*, the data instances. It is then processed by the subservient $\mathcal{SPARQL}$ engine

[53] https://jena.apache.org/documentation/rdf/
[54] https://jena.apache.org/documentation/ontology/
[55] https://www.youtube.com/watch?v=nM3dDy74qIw

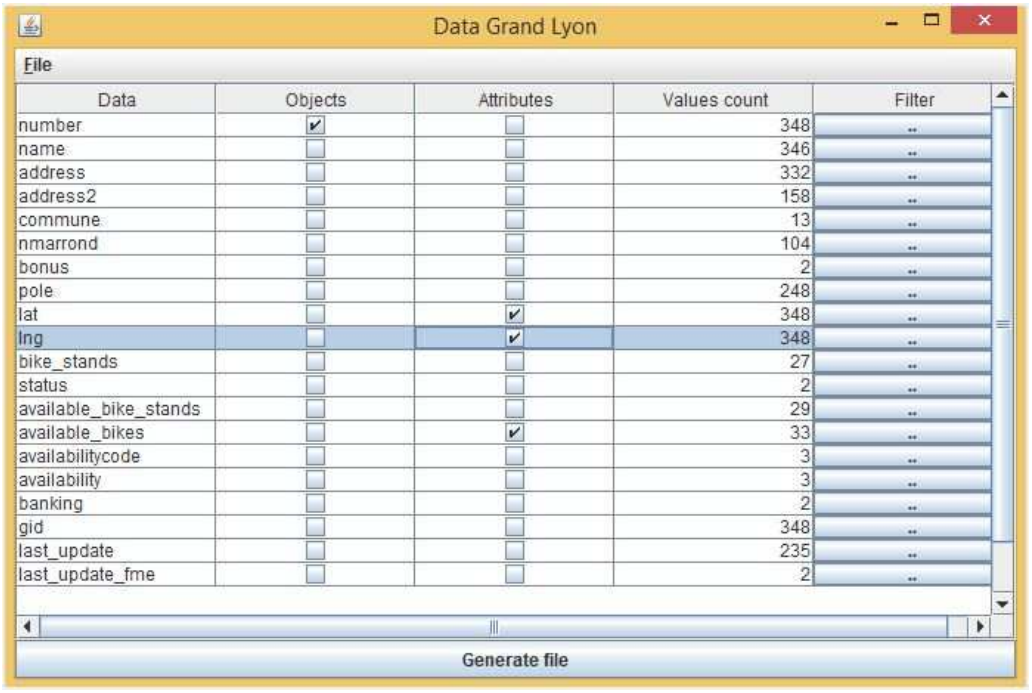

Figure 14: Velo'V object/attribute data selection from the Data Grand Lyon raw data

on the RDF-encoded data and the resulting sets of stations matching the normalized $\mathcal{OSF}$ query is returned; in this case, all the Velo'V stations found within 1000 meters and currently have more than 5 bikes available (see Figure 22). The result is then displayed in form intelligible to the user (see Figure 23).

Notation conventions for ontologies after approximations

We briefly explain the notational convention we adopted in the prototype for expressing the approximations specified by a user according to the type of approximations made in the Velo'V dataset from the Data Grand Lyon. Ontologies are created based on the approximations done on the dataset Data Grand Lyon as described in Section 4.4. These ontologies are used by the $\mathcal{CEDAR}$ Order-Sorted Feature ($\mathcal{OSF}$) reasoner and thus follow an $\mathcal{OSF}$ syntax.

The $\mathcal{OSF}$ ontologies created in the approximation phase have generated concept names corresponding to sets of approximated values (see for example Figure 20). They are of the form `Concept_0`, `Concept_1`, `Concept_2`, *etc.*, ... Attribute names and values are identifiers used in attribute declarations of the following form:

```
AttributeName__ApproximationValue >> ApproximationType
```

where `AttributeName`, `ApproximationValue`, and `ApproximationType` are strings

```
{"fields": ["number", "name", "address", "address2", "commune", "nmarrond", "bonus",
"nb_results": 348,
"values": [ ["6042", "Cit\u00e9 Internationale - R\u00e9sidence", "All\u00e9e Achille
"layer_name": "jcd_jcdecaux.jcdvelov",
"table_href": "https://download.data.grandlyon.com/ws/rdata/jcd_jcdecaux.jcdvelov.json",
"field_name": "gid"}
```


Figure 15: Example of the JSON appearance of the selected Velo'V data

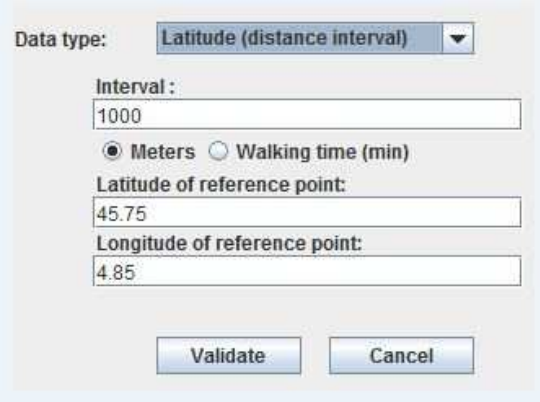

Figure 16: Example of the semantic zoom's GUI for setting the user's desired approximations in the Velo'V dataset

of characters defined as follows:

- `AttributeName` is the name of the attribute;
- `ApproximationType` is one of:
  - `NumericalThreshold`,
  - `NumericalInterval`,
  - `NumericalDifference`,
  - `DateDifference`,
  - `DistanceInterval`, or
  - `DistanceDifference`.

  Then, `ApproximationValue` depends of the chosen approximation type:
  - `NumericalThreshold`: when applying a numerical threshold approximation, the user is asked to enter a threshold. Them `ApproximationValue` has two possible values:

```
LM_VALUED_CONTEXT
| 6042 | 12002 | 10103 | 8059 | 10014 |
| position distance | available_bikes
| DistanceDifference=>difference=1250.0,
| non | sup
| non | sup
| non | sup
| non | sup
| non | sup
| non | sup
| non | sup
| non | inf
| non | inf
| non | inf
```

Figure 17: Example of the generated concept lattice in LMVT data format after desired level of semantic zoom

  - `inf`, which indicates that the value is inferior to the threshold; or,
  - `sup`, otherwise.

- `NumericalInterval`: when applying a Numerical Interval approximation, the user is asked to enter an interval (I). An ontology is created for each interval in the set $\{[0, I[, [I, 2I[, [2I - 3I[, \ldots\}$, each interval containing at least one value. Then, `ApproximationValue` is, respectively: `0-I`, `I-2I`, `2I-3I`, *etc.*, ...
- `NumericalDifference`: when applying a Numerical Difference approximation, the user is asked to enter a reference value (R), and a difference (D). A value V is considered close to the reference value if $|V - R| \leq D$. Then, `ApproximationValue` has two possible values :
  - `oui`: the value is close to the reference;
  - `non`: otherwise.
- `DateDifference`: when applying a Date Difference approximation, the user is asked to enter a start date and an end date. Then, `ApproximationValue` has two possible values :
  - `oui`: if the date is between the start date and the end dates;
  - `non`: otherwise.
- `DistanceInterval`: same as Numerical Interval, except that `AttributeName` is then `position_distance`.
- `DistanceDifference`: same as Numerical Difference, but `AttributeName` is `position_interval`.

```
1=>10001,10002,10004,10005,10006,10007,10008,1001,10011,10012,10013,
2=>10001,10002,10005,10006,10007,10008,10012,10013,10014,10018,1002,
3=>10004,1001,10011,10016,10021,10028,1003,10034,10041,10047,1005,10
4=>3004,3005,3009,3010,3015,3016,3018,3029,3031,3036,3037,3038,3039,
5=>10001,10002,10005,10006,10007,10008,10012,10013,10014,10018,1002,
6=>10004,1001,10011,10016,10021,10028,1003,10034,10041,10047,1005,10
7=>3004,3005,3010,3015,3016,3018,3031,3036,3037,3038,3058,3082,3084,
8=>3009,3029,3039,3053,3066,3067,3088,7004,7011,7014,7016,7023,7034,
```

Figure 18: Example of the text format associating a concept index to the ids of the values comprising this concept's extent as the result of semantic zooming

```
<rdf:RDF
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://data.grandlyon.com/"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#">
  <owl:Class rdf:about="http://data.grandlyon.com/DonneeGeographique"/>
  <owl:Class rdf:about="http://data.grandlyon.com/Station"/>
  <rdf:Property rdf:about="http://data.grandlyon.com/Position">
    <rdfs:range rdf:resource="http://data.grandlyon.com/DonneeGeographique"/>
    <rdfs:domain rdf:resource="http://data.grandlyon.com/Station"/>
  </rdf:Property>
  <owl:DatatypeProperty rdf:about="http://data.grandlyon.com/longitude">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
    <rdfs:domain rdf:resource="http://data.grandlyon.com/DonneeGeographique"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://data.grandlyon.com/lattitude">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#decimal"/>
    <rdfs:domain rdf:resource="http://data.grandlyon.com/DonneeGeographique"/>
  </owl:DatatypeProperty>
  <owl:DatatypeProperty rdf:about="http://data.grandlyon.com/Last_Update_fme">
    <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#date"/>
    <rdfs:domain rdf:resource="http://data.grandlyon.com/Station"/>
```

Figure 19: Example of RDF encoding of an $\mathcal{OSF}$ ontology using Semantic Web primitives (RDFS and OWL)

```
Concept_0, string, int, date, boolean, NumericalDifference, NumericalInterval,

Concept_1, Concept_2, Concept_3, Concept_4 <| Concept_0.
Concept_5, Concept_6 <| Concept_1.
Concept_5, Concept_7 <| Concept_2.
Concept_6, Concept_8 <| Concept_3.
Concept_7, Concept_8 <| Concept_4.


available_bikes__inf << Concept_3.
available_bikes__inf >> NumericalThreshold.


available_bikes__sup << Concept_2.
available_bikes__sup >> NumericalThreshold.


position_distance__non << Concept_1.
position_distance__non >> DistanceDifference.


position_distance__oui << Concept_4.
position_distance__oui >> DistanceDifference.
```

Figure 20: Ontology in Figures 17 and 19 in $\mathcal{OSF}$ syntax

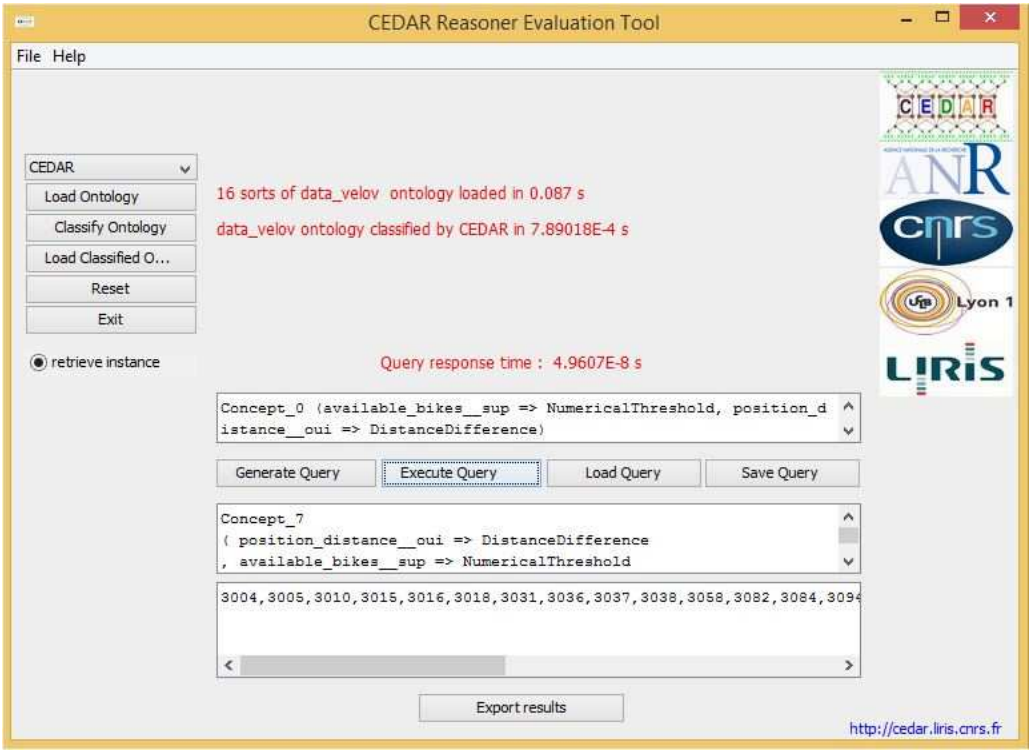

Figure 21: The *CEDAR* reasoner system processing the ontology of Figure 20

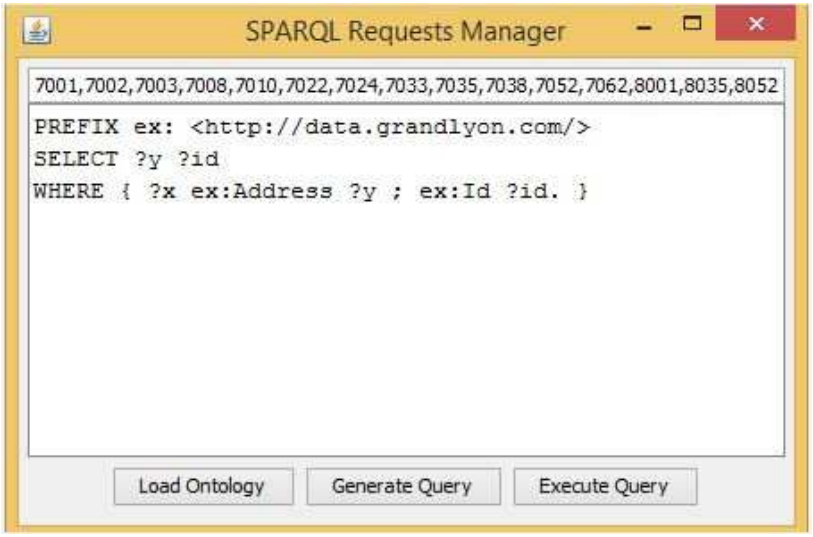

Figure 22: Our example's query in *SPARQL* form

Figure 23: Answer to the user query in Figure 22

References

[1] Hassan Aït-Kaci. $\mathcal{HOOT}$—a language for expressing and querying hierarchical ontologies, objects, and types—a specification. CEDAR Technical Report Number 16, $\mathcal{CEDAR}$ Project, LIRIS, *Département d'Informatique, Université Claude Bernard Lyon 1*, Villeurbanne, France, December 2014. [See online[56]].

[2] Hassan Aït-Kaci. Towards declarative scripting combining CP and analytics. Invited presentation, Workshop on Constraint Programming and Analytics (Panel), August 31, 2015. [See online[57]].

[3] Hassan Aït-Kaci and Samir Amir. Classifying and querying very large taxonomies with bit-vector encoding. *Journal of Intelligent Information Systems*, pages 1–25, September 2015. [See online[58]].

[4] Hassan Aït-Kaci and Philippe Bonnard. Handling uncertainty in a business-rule management system. Technical Paper, June 2013. [See online[59]].

[5] Mehwish Alam and Amedeo Napoli. Lattice-based views over SPARQL query results. In Ilaria Tiddi, Mathieu d'Aquin, and Nicolas Jay, editors, *Proceedings of the 1st Workshop on Linked Data for Knowledge Discovery*, pages 49–58, Nancy, France, September 19, 2013. CEUR-WS.org, CEUR Workshop Proceedings. [See online[60]].

[6] Samir Amir and Hassan Aït-Kaci. Design and implementation of an efficient semantic web reasoner. CEDAR Technical Report Number 12, $\mathcal{CEDAR}$ Project, LIRIS, *Département d'Informatique, Université Claude Bernard Lyon 1*, Villeurbanne, France, October 2014. [See online[61]].

[7] Peter Becker and Joachim Hereth Correia. The toscanaj suite for implementing conceptual information systems. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis, Foundations and Applications*, pages 324–348. Lecture Notes in Computer Science 3626, Springer, January 2005. [See online[62]].

[8] Stefano Bistarelli and Francesca Rossi. Semiring-based soft constraints. In Pierpaolo Degano, Rocco de Nicola, and José Meseguer, editors, *Concurrency, Graphs and Models—Essays Dedicated to Ugo Montanari on the Occasion of His 65th Birthday*, volume LNCS 5065, pages 155–173. Springer, 2008. [See online[63]].

[9] Christian Bizer, Tom Heath, and Tim Berners-Lee. Linked data – the story so far. *International Journal on Semantic Web and Information Systems*, 5(3):1–22, 2009. [See online[64]].

[10] Sofiene Chamakhi. Dérivation d'ontologies à partir de données brutes et zoom sémantique par analyse de concepts formels. Master's thesis, Département d'informatique, Université Claude Bernard Lyon 1, Villeurbanne, France, juin 2015. [See online[65]].

[11] Melisachew Wudage Chekol and Amedeo Napoli. An FCA framework for knowledge discovery in SPARQL query answers. In Eva Blomqvist and Tudor Groza, editors, *Proceedings of the 12th International Semantic Web Conference's Posters & Demonstrations Track, (ISWC 2013),*, pages 197–200, Sydney, Australia, October 23, 2013. CEUR-WS.org, CEUR Workshop Proceedings. [See online[66]].

[56] http://cedar.liris.cnrs.fr/papers/ctr16.pdf
[57] [...]_Towards_Declarative_Scripting_Combining_CP_and_Analytics
[58] http://hassan-ait-kaci.net/pdf/hak-sa-jiis2015.pdf
[59] http://cedar.liris.cnrs.fr/papers/BayesianConditionActionRules.pdf
[60] http://ceur-ws.org/Vol-1232/paper5.pdf
[61] http://cedar.liris.cnrs.fr/papers/ctr12.pdf
[62] http://www.researchgate.net/publication/221089330
[63] http://www.dmi.unipg.it/bista/papers/papers-download/paper4-last-revised3.pdf
[64] http://tomheath.com/papers/bizer-heath-berners-lee-ijswis-linked-data.pdf
[65] http://cedar.liris.cnrs.fr/palse/doc/CHAMAKHI_Sofiene_2014-2015_TIWE.pdf
[66] http://ceur-ws.org/Vol-1035/iswc2013_poster_5.pdf

[12] Philipp Cimiano, Andreas Hotho, and Steffen Staab. Learning concept hierarchies from text corpora using formal concept analysis. *Journal of Artificial Intelligence Research*, 24:305–339, August 2005. [See online[67]].

[13] Patrick Cousot. Abstract interpretation. *ACM Computing Surveys—Symposium on Models of Programming Languages and Computation*, 28(2):324–328, June 1996. Tutorial summary:[See online[68]].

[14] Patrick Cousot and Radhia Cousot. Abstract interpretation and application to logic programs. *Journal of Logic Pogramming*, 13(2-3):103–179, 1992.

[15] Olivier Curé and Robert Jeansoulin. An FCA-based solution for ontology mediation. *Journal of Computing Science and Engineering*, 3(2):90–108, June 2009. [See online[69]].

[16] Frithjof Dau and Barış Sertkaya. Formal concept analysis for qualitative data analysis over triple stores. In Olga De Troyer, Claudia Bauzer Medeiros, Roland Billen, Pierre Hallot, Alkis Simitsis, and Hans Van Mingroot, editors, *Proceedings of The 1st International Workshop on Modeling and Reasoning for Business Intelligence (MOREBI 2011)*, pages 45–54. LNCS 6999, Springer, November 2011. [See online[70]].

[17] Didier Dubois and Henri Prade. Fuzzy sets in approximate reasoning, part 1: Inference with possibility distributions. *Fuzzy Sets and Systems*, 40(1):143–202, March 1991. [See online[71]].

[18] Marcel Erné, Jürgen Koslowski, Austin Melton, and Geroge E. Strecker. A primer on Galois connections. In S. Andima *et al.*, editor, *Papers on General Topology and its Applications*, pages 103–125, New York, NY, USA, 1994. Annals of the Annals New York Academy of Science 704. [See online[72]].

[19] Bernhard Ganter and Rudolf Wille. *Formal Concept Analysis—Mathematical Foundations*. Springer, 1999.

[20] Hele-Mai Haav. A semi-automatic method to ontology design by using FCA. In Václav Snášel and Radim Bělohlávek, editors, *Proceedings of the 2nd International Workshop on Concept Lattices and their Applications*, pages 13–24, Ostrava, Czech Republic, September 2004. CLA 2004, CEUR-WS.org. [See online[73]].

[21] Markus Kirchberg, Erwin Leonardi, Yu Shyang Tan, Sebastian Link, Ryan K. L. Ko, and Bu-Sung Lee. Formal concept discovery in semantic web data. In Florent Domenach, Dmitry I. Ignatov, and Jonas Poelmans, editors, *Proceedings of the 10th International Conference Formal Concept Analysis, (ICFCA 2012)*, pages 164–179, Leuven, Belgium, May 7–10, 2012. LNCS 3626, Springer. [See online[74]].

[22] Uffe Kjærulff and Anders Madsen. Probabilistic networks—an introduction to Bayesian networks and influence diagrams, May 2005. [See online[75]].

[23] Zdzisław Pawlak. *Rough Sets*. Polish Academy of Science, Institute of Theoretical and Applied Informatics, Polish Academy of Sciences, Gliwice, Poland, 2004. [See online[76]].

[24] Laurent Perron. Google's OR tools. Invited presentation, Conference on Constraint Programming (CP 2013), September 2013. [See online[77]].

[67] http://www.aaai.org/Papers/JAIR/Vol24/JAIR-2409.pdf
[68] http://www.di.ens.fr/~cousot/AI/IntroAbsInt.html
[69] http://jcse.kiise.org/posting/3-2/jcse_3-2_45.pdf
[70] http://dr-dau.net/Papersneu/MoreBI_workshop_Dau_Sertkaya_final.pdf
[71] ftp://www.cc.uah.es/pub/.../Articulos/AproximateReasonning.pdf
[72] http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.36.6693
[73] http://ceur-ws.org/Vol-110/paper2.pdf
[74] http://slink.foiks.org/Articles/ICFCA12.pdf
[75] http://www.cs.aau.dk/~uk/papers/pgm-book-I-05.pdf
[76] http://bcpw.bg.pw.edu.pl/Content/2026/RoughSetsRep29.pdf
[77] http://cp2013.a4cp.org/sites/default/files/uploads/cpsolvers2013.ORTools.pdf

[25] Uta Priss. Linguistic applications of formal concept analysis. In Bernhard Ganter, Gerd Stumme, and Rudolf Wille, editors, *Formal Concept Analysis, Foundations and Applications*, pages 149–160. Springer, LNAI 3626, 2005. [See online[78]].

[26] Helmut Schmid. Probabilistic part-of-speech tagging using decision trees. In *Proceedings of International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, September 1994. [See online[79]].

[27] Helmut Schmid. LoPar: Design and implementation. *Arbeitspapiere des Sonderforschungsbereiches 340* 149, Institute for Computational Linguistics, University of Stuttgart, Stuttgart, Germany, July 2000. [See online[80]].

[28] Gerd Stumme and Alexander Maedche. FCA-MERGE: bottom-up merging of ontologies. In Bernhard Nebel, editor, *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 225–234, Seattle, Washington, USA, August 4–10 2001. Morgan Kaufmann. [See online[81]].

[29] Petko Valtchev, David Grosser, Cyril Roume, and Mohamed Rouane Hacene. Galicia: An open platform for lattices. In Peter M.A. Sloot, David Abramson, Alexander V. Bogdanov, Jack J. Dongarra, Albert Y. Zomaya, and Yuriy E. Gorbachev, editors, *Proceedings of the 11th International Conference on Conceptual Structures*, pages 241–254. ICCS 2003, Shaker Verlag, July 2003. [See online[82]].

Acknowledgements: The authors thank the LivEMUSIC Project's sponsors—*Université de Lyon*[83] and *Université Claude Bernard Lyon 1*[84]—for funding the project, as well as the LIRIS[85] for hosting the project.

They are also indebted to Prof. Marian Scuturici for advising them regarding access to the Data Grand Lyon platform in Task T1, as well as to Dr. Samir Amir for his kind help providing technical assistance for Task T3's use of the $\mathcal{CEDAR}$ reasoner.

[78] http://upriss.org.uk/papers/fcaic03.pdf
[79] http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/data/tree-tagger1.pdf
[80] http://www.cis.uni-muenchen.de/ schmid/papers/lopar.pdf
[81] http://www.kde.cs.uni-kassel.de/stumme/papers/2001/IJCAI01.pdf
[82] http://www.researchgate.net/publication/2907845

Project's Final Report

Living Environmnent Monitoring Use Scenario with Intelligent Control

Hassan Aït-Kaci, Sofiene Chamakhi, Tanguy Raynaud, Mathieu Veyrand

January 2016