# A Distributed Music Information System

Thesis submitted in partial fulfilment
of the requirements of the University of London
for the Degree of Doctor of Philosophy

**Yves Raimond**

Submitted: November 2008

Department of Electronic Engineering,
Queen Mary, University of London

I certify that this thesis, and the research to which it refers, are the product of my own work, and that any ideas or quotations from the work of other people, published or otherwise, are fully acknowledged in accordance with the standard referencing practices of the discipline. I acknowledge the helpful guidance and support of my supervisor, Prof Mark Sandler.

# Abstract

Information management is an important part of music technologies today, covering the management of public and personal collections, the construction of large editorial databases and the storage of music analysis results. The information management solutions that have emerged for these use-cases are still isolated from each other. The information one of these solutions manages does not benefit from the information another holds.

In this thesis, we develop a distributed music information system that aims at gathering music-related information held by multiple databases or applications. To this end, we use Semantic Web technologies to create a unified information environment. Web identifiers correspond to any items in the music domain: performance, artist, musical work, etc. These web identifiers have structured representations permitting sophisticated reuse by applications, and these representations can quote other web identifiers leading to more information.

We develop a formal ontology for the music domain. This ontology allows us to publish and interlink a wide range of structured music-related data on the Web. We develop an ontology evaluation methodology and use it to evaluate our music ontology. We develop a knowledge representation framework for combining structured web data and analysis tools to derive more information. We apply these different technologies to publish a large amount of pre-existing music-related datasets on the Web. We develop an algorithm to automatically relate such datasets among each other. We create automated music-related Semantic Web agents, able to aggregate musical resources, structured web data and music processing tools to derive and publish new information. Finally, we describe three of our applications using this distributed information environment. These applications deal with personal collection management, enhanced access to large audio streams available on the Web and music recommendation.

## Acknowledgements

I would like to thank my supervisor, Prof. Mark Sandler, for his constant support throughout my years in the C4DM group in Queen Mary, University of London. I would also like to thank the people who have been part of C4DM while I've been there. In no particular order: David P, Chris C, Chris H, Chris L, Matthew, Xue, Matthias, Adam, Louis, Juan, Katy, Amelie, Kurt, Emmanuel, Paul, Enrique, Dan S, Dan T, Steve, Andrew N, Andrew R, Mark P, Mark L, Beiming, Peyman, Robert, Nico, Gyorgy, Becky, Josh, Simon, Ivan and Maria.

I would especially like to thank Samer Abdallah for developing the original concepts underlying the Music Ontology framework. His ideas have had a profound impact on my work, and I will never thank him enough for that. I would also like to thank Christopher Sutton, for this amazing year we spent working together, and for taking the time to read this thesis. I also thank everybody in the Music Ontology community, especially Frederick Giasson, and in the Linking Open Data project, for their kindness and motivation, the Musicbrainz community for their outstanding work, as well as all the DBTune users for helping me spotting and fixing more bugs than I can count. I also thank the many people with whom I had the chance to work over these three years, in particular Oscar Celma, Alexandre Passant, Michael Hausenblas, Tom Heath, Chris Bizer, Wolfgang Halb and Francois Scharffe. I would also like to thank the people behind SWI-Prolog and ClioPatria, for developing the framework I used for most of the experiments described in this thesis. I would like to thank the BBC Audio & Music interactive team, for providing ideas, support and lots of data. I also acknowledge the support of a studentship from both the Department of Electronic Engineering and the Department of Computer Science at Queen Mary.

Finally, many thanks to my family for their continued support. I would especially like to thank Anne-Sophie for her constant love, support and understanding, even when I had to work overnight, on week-ends or on holidays.

# Contents

# List of Figures

# List of Tables

# Part I

# Introduction and background

# Chapter 1

# Introduction

The objective of this thesis is to make a contribution in the music information management research field. We create a distributed music information system, encompassing the following main aspects:

- Editorial – information about artists, releases, performances, recordings, musical works;
- Musicological – information about the actual content of a musical item, such as an harmonic progression or a segmentation in musically relevant segments;
- Workflows – know-how for deriving new information by combining music processing tools with existing information.

The information available within this information system can be used by automated agents. We create automated music processing agents, aggregating workflows and other music-related information to deduce and publish new information. We create other agents, performing specific tasks on behalf of a human user, for example to help organise a personal music collection.

## 1.1 Problem definition

Information management is an important part of music technologies today, covering the management of public and personal collections, the construction of large editorial databases and the storage of music analysis results. Software solutions to each of these problems have emerged, providing a myriad of tools — for example Greenstone[1] for digital libraries, iTunes[2] for personal media collection management, Musicbrainz[3] as a large online editorial database, and traditional relational databases for managing analysis results.

Despite these tools all working with music-related information, they are typically completely isolated from each other. Even between instances of the same tool, sharing and reusing information is a difficult task, often involving bespoke "glue code" and manual effort for one-time data migration. Common data formats reduce the need for such efforts, but restrict the expressivity of applications' data. This becomes a greater problem if we extend our range of interest to data which *could* be produced, for example by audio analysis algorithms providing higher-level representations than the audio signal itself.

[1]see `http://www.greenstone.org`
[2]see `http://www.iTunes.com`
[3]see `http://musicbrainz.org`

This thesis studies a distributed information system allowing us to integrate a vast array of music-related information sources.

A promising approach is to set out to build a music-related “web of data” which does not limit the formats or ontologies used to record information, allows arbitrary mixing and reuse of information by applications, and allows different information sources in different places to be interlinked. For example, an ethnomusicological archive may benefit from being linked to a geographical database. In this way, the archive can be tightly focused on its primary topic, and leave the burden of ancillary descriptions to other focused databases. The vision of the Semantic Web [BLHL01], and the vast array of technologies already established in pursuit of it, provide just the functionality required to begin building such a “web of data”.

## 1.2 Requirements

We consider the following requirements for building a music information system.

- Our system must be distributed. We must provide an abstract information space that can join up multiple data sources, located in multiple places.
- Our system must handle information from multiple origins. For example, information can originate from online databases or from music processing algorithms.
- Our system must handle a wide range of information needs. The needs of an ethnomusicological archive are different from the needs of an editorial database.
- Music-related information must be explicit. Machines must be able to process and integrate it.
- Music-related information must be given well-defined meaning. Machines must be able to meaningfully interpret it.
- If music-related information is explicit and has well-defined meaning, it can be automatically processed to deduce new information. We can create automated music processing agents, deducing new information and making it available.

## 1.3 Research method

> “Two paradigms characterise much of the research in the Information Systems discipline: behavioural science and design science. The behavioural-science paradigm seeks to develop and verify theories that explain or predict human of organisational behaviour. The design-science paradigm seeks to extend the boundaries of human and organisational capabilities by creating new and innovative artifacts. [...] Artifacts are broadly defined as *constructs* (vocabulary and symbols), *models* (abstractions and representations), *methods* (algorithms and practises), and *instantiations* (implemented and prototype systems).” [HMP+04]

This thesis follows the design-science research method. We develop an artifact for managing music-related information in a distributed context. The two main parts of this thesis describe different aspects of this artifact.

### 1.3.1 Constructs and models

We develop constructs and models supporting music-related information in a distributed context in Part II. This leads us to develop two knowledge representation frameworks: the Music Ontology framework for editorial and musicological information, and the N3-Tr framework for music analysis workflows. The novelty of these frameworks is shown by comparing them with related works. We evaluate the following aspects [JGPP95] of our representation frameworks:

- *Verification.* We ensure that they are defined correctly and that they function correctly.
- *Validation.* We ensure that they model the domain for which they were created.
- *Assessment.* We ensure that they are useful to solve real-world problems. Evidence of their utility is provided in Part III.

### 1.3.2 Methods and instantiations

We develop methods and instantiations in Part III. We detail how we used the Music Ontology and the N3-Tr representation frameworks to gather a wide range of music-related information in a single distributed system: the Web. We also detail novel applications making use of the information held by this system.

## 1.4 Thesis outline

The core of this thesis is divided in two main parts. Part II describes our distributed music information system. Part III focuses on applying our system to produce a large and dynamic music-related information environment, and using this new environment in novel applications. In the following, the thesis is outlined by summarising each chapter.

**Part I: Introduction and background**

**Chapter 1: Introduction.** We introduce and state the problem studied in this thesis. Music-related information is currently managed by heterogeneous systems which are isolated from each other. We study a distributed information system allowing us to integrate a vast array of music-related information sources.

**Chapter 2: Knowledge Representation and Semantic Web technologies.** We review the areas which build the foundations of our research. We focus here on Knowledge Representation and Semantic Web technologies. We give more focused background at the beginning of chapters 3, 4 and 5. We begin each of these chapters with a review and a discussion of the state of the art in the corresponding research domains, respectively music-related knowledge representation, ontology evaluation, and knowledge representation and management for multimedia processing workflows.

**Part II: A web-based music information system**

**Chapter 3: Conceptualisation of music-related information.** In order to publish and interlink structured music-related data on the Web, we need an ontological framework specifying how the music domain can be structured. This ontological framework provides a set of anchor points, on which music-related information can be hooked. To this end, this chapter develops the Music Ontology framework. This ontology is able to express a wide range of complex editorial information and musicological information. We evaluate how wide this range of information is in chapter 4.

**Chapter 4: Evaluation of the Music Ontology framework.** This chapter describes our ontology evaluation methodology, using as a basis a set of real-world music-related user needs and evaluating how well a system backed by an ontological framework could be used to answer these needs. We evaluate our Music Ontology framework using this methodology, and reach a measure for the range of music-related information it covers.

**Chapter 5: Music Processing Workflows on the Web.** Now we can publish a large range of structured music-related data on the Web, we are able to perform automated tasks on it. Such tasks can themselves derive new information out of existing information, which can be re-used by other tasks. This chapter details a logic and a corresponding syntax for music analysis workflows, combining structured web data and analysis tools to derive new structured web data. All this defines our N3-Tr framework. This new data can be given *meaning* by relating it to terms within our ontological framework. This chapter also details a publishing mechanism, allowing such derived information to be made available on the Web. The derived data can then be reused by further workflows or applications.

**Part III: Applications**

**Chapter 6: A web of music-related data.** The Music Ontology framework can be applied to publish and interlink a wide range of structured music-related data. This chapter describes our publication of a wide range of music-related datasets and our algorithm for automatically interlinking such datasets. This leads us to the creation of an integrated music-related information environment.

**Chapter 7: Automated music processing agents.** Structured web data and N3-Tr workflows can be aggregated and interpreted to derive new data. This leads to the creation of automated music-related web agents, aggregating audio content, information surrounding this content and analysis tools to publish new data. This chapter details an implementation of such an automated agent, as well as an available instance.

**Chapter 8: Case studies.** We now have in a single distributed environment a wide range of music-related information. This chapter describes novel applications using this information, in three contexts: personal collection management, enhanced access to syndicated audio content and music recommendations.

**Part IV: Conclusion and future work** We summarise the contributions of this thesis and outline directions for future work.

## 1.5 Major contributions

This thesis has the following major contributions:

1. A rich and expressive ontology for music-related information [RASG07], covering both complex editorial information and musicological information, that is being extended and revised by a significant community of researchers and practitioners.
2. An expression [GR07] of this ontology as a web ontology, using OWL (Web Ontology Language). Access statistics for the corresponding online specification from January 2007 to June 2008 are depicted in Figure 1.1.
3. A method for ontology evaluation against real-world user queries.
4. An evaluation of the Music Ontology framework.
5. A formalisation of Concurrent Transaction Logic in a Semantic Web context.
6. A representation framework for music analysis workflows: N3-Tr.
7. A method to publish information generated on-demand on the Web [RSS09].
8. A set of tools for publishing structured data on the Web.
9. A large web of structured music-related data [RS08, BHAR07].
10. An algorithm for automatically interlinking information sources on the Web [RSS08].
11. An automated tool interpreting structured web data and N3-Tr workflow to derive new information [RS08].
12. A set of applications illustrating the use of this large web of music-related information, dealing with personal music collections [RSS08], audio content on the Web [CR08] and music recommendations [PR08].

## 1.6 Published Work

Parts of the work presented in this thesis have been published in international journals and in the proceedings of international conferences and refereed workshops. The corresponding publications are listed below. We also include a reference to the parts of this thesis based on the corresponding work.

### 1.6.1 International Journals

- Y. Raimond, C. Sutton, and M. Sandler, "Publishing and interlinking music-related data on the Web," *Accepted for publication in IEEE Multimedia*, April-June 2009. The methodology described in this paper for publishing structured music-related web data derived by music analysis algorithms is detailed in § 5.3.
- O. Celma and Y. Raimond, "Zempod: A Semantic Web approach to podcasting," *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 6, pp. 162–169, March 2008. The system described in this paper is further expanded in § 8.2. In particular, we focus in this thesis on the N3-Tr aspects of this system.

### 1.6.2 International Conferences

- Y. Raimond and M. Sandler, "A web of musical information," in *Proceedings of the International Conference on Music Information Retrieval*, (Philadelphia, USA), 2008. The different interlinked music-related web datasets mentioned in this paper are further described in § 6.3 and § 7.3.

- M. Hausenblas, W. Halb, Y. Raimond, and T. Heath, "What is the size of the Semantic Web?," in *International Conference on Semantic Systems (I-SEMANTICS)*, 2008. The results described in this paper are reproduced in § 6.5.

- T. Scott, Y. Raimond, P. Sinclair, and N. Humfrey, "The Programmes Ontology," in *Proceedings of the XTech conference*, May 2008. We briefly mention the Programmes Ontology in § 6.3.3.

- Y. Raimond, S. Abdallah, M. Sandler, and F. Giasson, "The Music Ontology," in *Proceedings of the International Conference on Music Information Retrieval*, pp. 417–422, September 2007. A more detailed description of the Music Ontology framework is given in § 3.3.

- Y. Raimond and M. Sandler, "Using the Semantic Web for enhanced audio experiences," in *Proceedings of the 123rd Audio Engineering Society convention*, 2007. The personal music collection management software mentioned in this paper is further described in § 8.1.

- C. Bizer, T. Heath, D. Ayers, and Y. Raimond, "Interlinking open data on the web," in *Demonstrations Track, 4th European Semantic Web Conference, Innsbruck, Austria*, 2007. We give a brief overview of the Linking Open Data project in § 6.1.

- Y. Raimond, S. Abdallah, M. Sandler, and M. Lalmas, "A scalable framework for multimedia knowledge management," in *Proceedings of the 1st conference on Semantic and Digital Media Technologies* (Springer, ed.), Lecture Notes in Computer Science, December 2006. The Semantic Web part of the framework described in this paper form the basis of the work detailed in chapter 5. In particular, the 'Knowledge Machines' described in this paper are similar to Semantic Web agents interpreting N3-Tr workflows, such as the agent described in chapter 7.

- S. Abdallah, Y. Raimond, and M. Sandler, "An ontology-based approach to information management for music analysis systems," in *Proceedings of 120th AES convention*, 2006. The work described in this paper form the basis of the work on the Music Ontology described in § 3.3 and the work on a logic-based representation of music analysis workflows described in § 5.2.

### 1.6.3 Refereed Workshops

- A. Passant and Y. Raimond, "Combining social music and Semantic Web for music-related recommender systems," in *Proceedings of the Social Data on the Web workshop*, (Karlsruhe, Germany), October 2008. This work is described in § 8.3.

- Y. Raimond, C. Sutton, and M. Sandler, "Automatic interlinking of music datasets on the Semantic Web," in *Proceedings of the Linked Data on the Web workshop, colocated with*

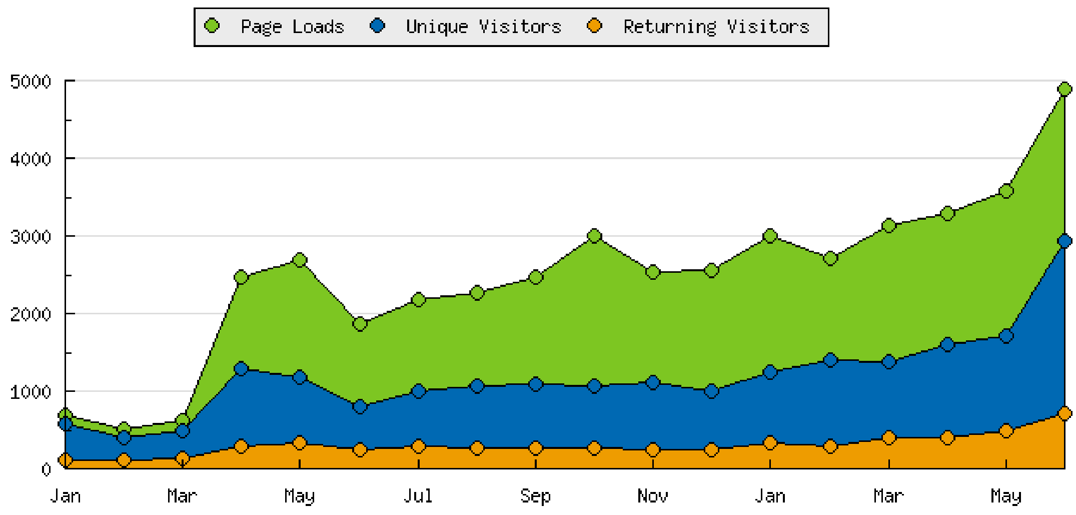


Figure 1.1: Access statistics for the Music Ontology online specification, from January 2007 to June 2008

*the World Wide Web Conference*, 2008. The automated interlinking algorithm described in this paper is further detailed in § 6.4. In particular, we provide further evaluations, and compare our approach with related work.

- Y. Raimond, C. Sutton, and M. Sandler, "A distributed data space for music-related information," in *Proceedings of the International Workshop on the Many Faces of Multimedia Semantics, colocated with ACM-Multimedia*, 2007. The publication mechanism described in this paper is further detailed in § 5.3.
- F. Scharffe, Y. Raimond, L. Barthelemy, Y. Ding, and M. Luger, "Publishing and accessing digital archives using the EASAIER framework," in *Proceedings of the First International Workshop on Cultural Heritage on the Semantic Web, co-located with the International Semantic Web Conference, Busan, Korea*, 2007. We briefly mention the EASAIER project in § 8.4.

# Chapter 2

# Knowledge Representation and Semantic Web technologies

First, we define three terms that we will be using throughout this thesis: data, information and knowledge [AN95]:

- Data – Data is a set of symbols. Data has no meaning in itself, it is purely syntactic. For example, an array of numbers and a set of characters are data.
- Information – Information is data that has been given meaning by way of relational connections. For example, the relation 'this array of numbers is the spectrogram of that audio signal' constitutes information. We also use the term 'structured data' in this thesis, which characterises these relational connections between pieces of data — structured data is information.
- Knowledge – Knowledge is information that has been incorporated in an agent's reasoning process. It is the output of a learning step. A simple form of learning is the act of aggregating information.

In this chapter, we review the areas which constitute the foundations of our research. We review two main areas, knowledge representation in § 2.1 and web technologies in § 2.2. More detailed and focused background is given at the beginning of the next thesis chapters.

## 2.1 Knowledge Representation

In order for machines to help with knowledge management issues, we need a framework in which knowledge can be made explicit. Such a framework allows machines to automatically process knowledge, and to share knowledge unambiguously. The basic problem of knowledge representation is then the development of a sufficiently precise notation for representing knowledge [RN95].

We review in this section a number of knowledge representation frameworks that we will use throughout this thesis.

### 2.1.1 Propositional logic

The propositional logic provides a formal mechanism for reasoning about statements built using atomic propositions (constants) and logical connectives. An atomic proposition is a symbol, e.g. $a$ or $b$, standing for something which may be true or false, e.g. 'this performance was conducted by that person'. The logical connectives $\vee$ (*or*), $\wedge$ (*and*), $\neg$ (*not*), $\equiv$ (*equivalent to*) and $\supset$ (*implies*, where $a \supset b$ is equivalent to $b \vee \neg a$) can be used to build composite formulæ (sentences), such as $\neg a \vee b$ or $a \supset b$.

### 2.1.2 First-Order Logic

The propositional logic is rather limited in the sort of knowledge it can represent, because the internal structure of the atomic propositions is hidden from the logic. For example, the atomic proposition 'this performance was conducted by that person' could be rewritten 'xyz' — it is impossible to derive that this proposition relates a performance event and a person.

#### 2.1.2.1 FOL syntax

The First-Order Logic (FOL) extends the propositional logic by introducing a way to express statements about constant symbols using predicates, which are essentially parametrised propositions. For example, if we consider the symbols $a$ and $b$, we can express the atomic proposition '$a$ was conducted by $b$' using the binary predicate $\mathsf{conductor}(a, b)$. The FOL syntax also introduces another set of symbols, called functions. Functions of arity 0 are constant symbols. A function of arity 2 is for example $+(a, b)$, corresponding to the sum of $a$ and $b$.

Finally, FOL introduces variables as well as existential ($\exists$) and universal ($\forall$) quantification. For example, we can express that all persons who conducted a performance are conductors.

$$\forall \mathsf{Person}.\ \exists \mathsf{Performance}.\ \mathsf{conductor}(\mathsf{Performance}, \mathsf{Person}) \supset \mathsf{Conductor}(\mathsf{Person})$$

#### 2.1.2.2 FOL semantics

The semantics of FOL are given by Tarski's model theory [Hod93], referred to as model-theoretic or denotational semantics. FOL sentences are true or false with regard to a model, which consists of a domain of discourse $\Delta$ (a non-empty set of arbitrary entities) and an interpretation function $\cdot^{\mathcal{I}}$ assigning each constant symbol to a domain element, each predicate symbol to a relation on the domain and each function symbol to a function on the domain:

- If $a$ is a constant symbol, then $a^{\mathcal{I}} \in \Delta$;
- If $p$ is a $n$-ary predicate symbol, then $p^{\mathcal{I}} \in \Delta^n$;
- If $f$ is a $n$-ary function symbol, then $f^{\mathcal{I}} \in [\Delta^n \to \Delta]$, where a function $f : \mathcal{D} \to \mathcal{R}$ is a relation $f \subseteq \mathcal{D} \times \mathcal{R}$ such that if $(x, y) \in f$ and $(x, z) \in f$ then $y = z$.

The *truth* of a FOL predicate is then defined with regard to such a model.

**Definition 1.** A predicate $\mathsf{pred}(\mathsf{term}_1, \mathsf{term}_2, \cdots, \mathsf{term}_n)$, where $\mathsf{term}_i$ is either a variable, a constant or a function, is true in a certain model iff the domain elements that are the interpretation of $\mathsf{term}_1, \cdots, \mathsf{term}_n$ are in the relation that is the interpretation of $\mathsf{pred}$.

For example, if we consider the domain $\Delta = \{e_1, e_2, \ldots, e_n\} \cup \{p_1, p_2, \ldots, p_m\}$, and $a^{\mathcal{I}} = e_1$, $b^{\mathcal{I}} = p_2$, $c^{\mathcal{I}} = e_2$, $\mathsf{conductor}^{I} = \{(e_1, p_2)\}$, then $\mathsf{conductor}(a, b)$ is true and $\mathsf{conductor}(c, b)$ is not true.

The semantics of the universal and existential quantification are defined as follows.

**Definition 2.** $\forall X.\ S$ is true iff for all of the domain elements $e$, $S$ is true in the model where $X$ is interpreted by $e$ (we call such an association a *variable assignment*).

**Definition 3.** $\exists X.\ S$ is true iff there is a domain element $e$ such that $S$ is true in the model where $X$ is interpreted by $e$.

For example, $\exists P.\mathsf{conductor}(P, b)$ is true iff $b$ conducted something.

We now define the notions of *validity* and *satisfiability.*

**Definition 4.** A FOL sentence is valid iff it is true in all its interpretations.

**Definition 5.** A FOL sentence is satisfiable iff it is true in one of its interpretations.

#### 2.1.2.3 Entailment and proof

A collection of FOL sentences constitutes a knowledge base. Given a knowledge base $KB$, if some sentence $\phi$ is true in every interpretation in which all the sentences in $KB$ are true, then we say that $KB$ entails $\phi$, written $KB \models \phi$. The entailment therefore holds at a semantic level. A theory $\mathcal{T}$ is a set of sentences that are closed under entailment. For all $\phi$ such that $\mathcal{T} \models \phi$, $\phi \in \mathcal{T}$.

An inference procedure is an algorithm that computes on the symbolic level the sentences that are entailed by the knowledge base. Given a set of sentences $\psi$ and a given inference procedure, if some sentence $\phi$ is provable, then we write $\psi \vdash \phi$. Two desirable properties of an inference procedure are:

- Soundness – Everything provable is true.

$$s \models p \supset (p \vdash q \supset s \models q)$$

- Completeness – Everything true is provable.

$$(s \models p \supset s \models q) \supset p \vdash q$$

The Gödel Completeness Theorem [G2̈9] establishes the correspondence between truth and provability in FOL. A sentence $\phi$ is valid iff $\phi$ is provable.

FOL has a sound and complete inference procedure called resolution refutation. However, refutation is intractable, therefore making it difficult to use on large knowledge bases.

#### 2.1.2.4 Inference mechanisms

Therefore, when implementing an inference procedure, some mechanisms are applied to make it practical. Two main approaches are used for performing this task. In the forward-chaining approach, we focus on the *Modus ponens* rule of inference $(p \wedge (p \supset q) \vdash q)$. When new sentences are inserted in the knowledge base, we trigger this rule and derive new sentences. Examples of forward-chaining systems are CWM[1] and the Jess forward-chaining engine[2]. In the backward-chaining approach, we focus on the *Modus tollens* rule of inference ($\neg q \wedge (p \supset q) \vdash \neg p$). Backward-chaining is used at querying time. When questions are posed to the system, it tries to answer them using its knowledge base. Examples of backward-chaining systems are the inference engine we detail in chapter 7 and the Prolog programming language [CM84]. One of the main differences between Logic Programming and FOL is that the former is based on a Closed World Assumption instead of an Open World Assumption. In the Closed World Assumption, what is not known as true is false. In the Open World Assumption, what is not known as true is unknown. The Open World Assumption in FOL is closely related to its monotonicity, i.e. adding new statements does not change earlier positive or negative conclusions.

There are other inference types than deduction, such as abduction, where we try to generate an explanation for a set of sentences, or induction, where we try to generalise a set of sentences. However, both of them are not valid inference paradigms. The derived sentences constitute some hypotheses rather than true statements.

### 2.1.3 Higher-order logic and reification

Several extensions of FOL exist, such as Second Order Logic, which includes all possible relations among simple individuals in its domain of discourse. This permits, for example, to write sentences about sentences, e.g. " "$a$ was conducted by $b$" is believed by $c$". Higher Order Logic also includes all possible relations of relations etc. in its domain.

Some higher-order logic features can however be included in FOL. The *reification* process consists of creating a constant corresponding to predicates. These constants can then be used in further sentences. Since they are constants, the resulting theory is still within FOL. Reification can be used to express statements about statements. For example, we associate the predicate $\mathsf{conductor}(a, b)$ with the constant $p$. Then, we can use $p$ in other sentences, such as $\mathsf{believed\_by}(p, c)$. Further higher-order logic capabilities can be tackled within the FOL scope by stratifying the logic in two FOL levels, the object and the meta level [Kow79]. Frame Logic [KLW95] is an example of a logic combining FOL semantics with higher-order syntax.

### 2.1.4 Semi-decidability of FOL

FOL is only semi-decidable [Chu36]. There is a procedure which, for any arbitrary sentence, will halt iff the sentence is true. However, if the sentence is false, there is no guarantee that a procedure will be able to determine this. The procedure may never halt. Unlike in propositional logic, there is no decision procedure in FOL that determines whether an arbitrary sentence is valid or not. There is therefore a need for a decidable fragment of FOL.

### 2.1.5 The need for ontologies

A large part of building a logic-based information system is deciding the type of objects and the different relations that are going to be relevant in the system's domain. Designing an ontology of the domain involves identifying the important concepts and relations in a domain, and as such

[1]See `http://www.w3.org/2000/10/swap/doc/cwm.html`
[2]See `http://jessrules.com/jess/`

help to bring some order to the potentially chaotic collection of predicates that could be defined. Tackling these problems involves the definition of an *ontology* [Gru93], specifying the important concepts and relationships in a domain.

### 2.1.6 Description Logics

Description Logics (DL [BCM$^+$03]) are a family of sub-languages of FOL. These languages allow to capture the different concepts, relationships and individuals in a particular domain. A DL is composed of the following alphabets of symbols, along with their interpretation:

- Constants, interpreted as individuals in the domain.
- Atomic concepts, designated by unary predicate symbols. Concepts are given a set-theoretic interpretation: they are interpreted as a set of individuals.
- Atomic roles (or properties), designated by binary predicate symbols. Roles are interpreted as sets of pairs of individuals.

We also consider two concepts. The top concept $\top$ is interpreted as the set of all individuals. The bottom concept $\perp$ is interpreted as the empty set.

Sentences are then built from these basic symbols using a set of constructors. The available constructors vary from one DL to the other, as illustrated in [GPFLC04, p. 15]. We give in table 2.1 different examples of constructors available in the $\mathcal{SHOIN}(D)$ DL[3] that we use in chapter 3.

A number of inferences can be performed on a set of sentences built using these constructors. For example, we can perform:

- Subsumption – We check whether a concept always denotes a subset of the set denoted by a second concept.
- Satisfiability – We check whether a given concept expression is not necessarily interpreted as the empty set (this is a particular case of subsumption reasoning).
- Consistency – We verify that all concepts are satisfiable.
- Realisation – We find the most specific concept an individual belongs to.
- Retrieval – We find all individuals belonging to a given concept.
- Instance checking – We verify whether a given individual belongs to a given concept. All the previous inferences can be defined in terms of this inference.

The decidability and the complexity of the inference problem depend on the expressive power of the Description Logic. The $\mathcal{SHOIN}(D)$ description logic used in this thesis is a trade-off between expressiveness and decidability — there exists a decision procedure which terminates, both for positive and negative answers, unlike in FOL.

[3]The $(D)$ specifies that datatype properties can be used, e.g. a property relating a person to a string corresponding to his name.

| Constructor | DL | FOL | Example |
|---|---|---|---|
| Classification | $i : C$ | $C(i)$ | $i$ : $\mathsf{Conductor}$ specifies that $i$ is a conductor |
| Relationship | $< i_1, i_2 >: P$ | $P(i_1, i_2)$ | $< e, p >$: $\mathsf{conductor}$ specifies that $p$ is the conductor of $e$ |
| Conjunction | $C_1 \sqcap \ldots \sqcap C_n$ | $C_1(x) \wedge \ldots \wedge C_n(x)$ | $\mathsf{Percussion} \sqcap \mathsf{String}$ corresponds to the set of instruments that are both percussion instruments and string instruments |
| Disjunction | $C_1 \sqcup \ldots \sqcup C_n$ | $C_1(x) \vee \ldots \vee C_n(x)$ | $\mathsf{Percussion} \sqcup \mathsf{String}$ corresponds to the set of instruments that are either percussion instruments or string instruments |
| Complement | $\neg C$ | $\neg C(x)$ | $\neg\mathsf{Conductor}$ is the set of all individuals that are not conductors |
| Universal restriction | $\forall P.C$ | $\forall y.P(x, y) \supset C(y)$ | $\forall\mathsf{performer.Japanese}$ corresponds to the set of individuals that have been performed by only Japanese persons |
| Existential restriction | $\exists P.C$ | $\exists y.P(x, y) \wedge C(y)$ | $\exists\mathsf{conducted.Performance}$ corresponds to the set of all individuals that conducted a performance |
| Subsumption | $C \sqsubseteq D$ | $\forall x.C(x) \supset D(x)$ | $\mathsf{Piano} \sqsubseteq \mathsf{String}$ specifies that all pianos are string instruments |
| Role hierarchy | $P \sqsubseteq Q$ | $\forall x, y.P(x, y) \supset Q(x, y)$ | $\mathsf{violinist} \sqsubseteq \mathsf{performer}$ specifies that if someone is in the violinist relationship with a performance, he is also in the performer relationship |
| Inverse role | $Q \equiv P^-$ | $\forall x, y.P(x, y) \equiv Q(y, x)$ | $\mathsf{conductor} \equiv \mathsf{conducted}^-$ specifies that a person conducted a performance iff the performance was conducted by him |
| Transitive role | $P^+ \sqsubseteq P$ | $\forall x, y, z.P(x, y) \wedge P(y, z) \supset P(x, z)$ | $\mathsf{before}^+ \sqsubseteq \mathsf{before}$ specifies that if $a$ is before $b$ which is before $c$, then $a$ is before $c$ |
| Functional role | $\top \sqsubseteq \leq 1P.\top$ | $\forall x, y, z.P(x, y) \wedge P(x, z) \supset y = z$ | $\top \sqsubseteq \leq 1\mathsf{dft}.\top$ specifies that two Discrete Fourier Transform derived from a single signal will be the same |
| Collection of individuals | $\{a_1, a_2, \ldots, a_n\}$ | $x = a_1 \vee x = a_2 \vee \ldots \vee x = a_n$ | $\{\mathsf{myflute}, \mathsf{mypiano}, \mathsf{myguitar}\}$ specifies a set composed of three individuals |

Table 2.1: Constructors in the $\mathcal{SHOIN}(D)$ DL, along with DL syntax, FOL translation using the mapping in [GHVD03] and example

## 2.2 Web technologies

We focus in this section on technologies that we will use throughout this thesis, established in pursuit of Tim Berners-Lee's vision of the Semantic Web [BLHL01]. These technologies will be used as the starting point to explore the development of new solutions.

### 2.2.1 Introduction

The World Wide Web was first created for integrating many disparate information systems [BL89], based on ideas by Vannevar Bush [Bus45] and Douglas Engelbart [EE68]. Different sorts of computers existed, running different operating systems and using different programs. It took a long way to get from the information stored on a computer to related information on another computer, even if both computers were available on a single network. The approach of the Web was to include all of this information in an abstract space. Information on the Web can be located on different computers, maintained by different people, in different parts of the world. Such information can refer to other entities in the Web space. Therefore, one can jump seamlessly from one piece of information to another in this abstract space. The Web abstracts the network to create a unified information space.

Over the last years, the Web established itself as a unified interface to the Internet computer network. It contains interlinked multimedia resources concerning almost every imaginable subjects, and all that is accessible to anyone with an Internet connection. This success is mainly due to the fact that web technologies are simple, and do not enforce strong constraints — for example, it is possible to refer to non-existing information.

### 2.2.2 Towards the Semantic Web

However, the World Wide Web had a potential that was untapped. The Web got mainly used to create a large and distributed hypertext document, but machines were unable to process and integrate the information it holds meaningfully. The problem was similar to the one described to motivate the transition from propositional logic to first-order logic in § 2.1.2. There is no way to investigate the different symbols (e.g. documents) available on the Web. This limitation makes it impossible to ask questions to the Web (e.g. 'who performed that musical work?'), impossible to use web information to automate tasks (e.g. 'find gigs by artists similar to my most-played artists which fit within my vacation plan') and impossible to visualise and access information in multiple ways (e.g. 'display me the wave-form along with a segmentation in chorus, verses, etc, for that audio signal').

Recent work on web mining tools, extracting information from web documents, tackles this issue. For example, Schedl [Sch08] describes a web mining framework for extracting information about music artists from web documents. Also, web information retrieval algorithms exploit the graph structure of interlinked web documents to improve search results. The more a document is being linked to, the more it can be considered as relevant. The link structure is then used to learn more about a particular document.

However, these approaches have limitations. No information extraction algorithm is robust enough to handle multiple domains, languages and media (e.g. text, audio and video), and the graph structure of interlinked web documents still does not help to investigate the information held within a document, it just provides a context for the document as a whole.

Another approach is to make information *explicit* on the Web, therefore allowing machines to process and integrate it. We then create a Semantic Web:

> "The Semantic Web is not a separate web but an extension of the current web, in which information is given well-defined meaning, better enabling computers and people to work in cooperation." [BLHL01]

This approach is further motivated by the fact that a large amount of web sites are driven by relational databases. The information held within these databases is represented as web documents. An information extraction process is then the inverse process: extracting the information out of the representation. By making information explicit on the Web, we bypass this lossy step, and directly make the actual information available. This approach is also motivated by the fact that, even after running an information extraction process, we still need a way to make this information available explicitly in order for machines to be able to reuse it.

### 2.2.3 A web of data

The Web is built around the concept of URI (Uniform Resource Identifier[4]). A URI can identify anything, from a document to a person, a performance, an audio signal, etc. Web resources can have multiple associated representations. For example, to a URI identifying a particular signal, we may want to associate an HTML representation of it (providing some human-readable information about its characteristics) or an image depicting the actual waveform. The HTTP protocol[5] provides a common access mechanism for web resources. The HTTP GET operation[6] allows us to get from a web identifier to the corresponding representations. The Accept HTTP header allows us to specify what type of representation we want to get back. The web aspect comes into place when other web identifiers are mentioned within such representations. For example, the HTML representation of our signal URI might link to a URI identifying the corresponding recording device.

Now, these representations can be *structured*: they can provide explicit machine-processable information. When such representations quote other resource identifiers, enabling access to corresponding structured representation, we create a 'web of data'. This allows dynamic exploration of linked data sets [BL06a] which may be distributed across geographic locations, institutions and owners like documents on the document web.

### 2.2.4 Machine-processable representations

We now review different technologies for machine-processable representations of web identifiers.

#### 2.2.4.1 XML

The Extensible Markup Language (XML[7]) allows angle-bracketed tags to be embedded in text data, to provide additional information about the text. XML defines a meta-language: it can be used to define other languages, by defining a set of structural rules — what tags can be used,

[4]`http://www.ietf.org/rfc/rfc2396.txt`

[5]`http://www.w3.org/Protocols/rfc2616/rfc2616.html`

[6]along with 303 (See Other) redirects, in order to distinguish a web resource identifying the thing (e.g. the signal) from web resources identifying its representations [SC08]

[7]`http://www.w3.org/XML/`

and how they can be nested in each other. XML then provides a way to automatically determine whether a document conforms to one of these languages. An example XML document is the following one, in which we mention the title of an album, the name of the artist, and the titles of three of its tracks.

```
<?xml version="1.0" encoding="UTF-8"?>
<album id="33">
  <artist>Both</artist>
  <title>Simple Exercice</title>
  <tracks>
    <track>Simple Exercice</track>
    <track>The Monster</track>
    <track>Qui Dechante</track>
  <tracks>
</album>
```

XML is very flexible, and can be used to represent any arbitrary information. However, this flexibility is problematic for machine processing. Tags cannot be attached to some meaning, so XML processing applications usually just handle a well-defined set of tags, whose meaning is defined in natural language. Only the XML syntax can be automatically and generically processed. This lack of semantics makes it hard to process and integrate information which may be serialised in a number of ways in different XML documents.

One way of tackling this issue is to associate machine-processable meaning with the tags using the knowledge representation techniques described in § 2.1, which is the purpose of RDF and web ontologies.

#### 2.2.4.2 The RDF model

RDF originates from the Platform for Internet Content Selection (PICS[8]), a standard for associating information with content available on the Internet. RDF has been developed by two consecutive working groups within the World-Wide Web Consortium (W3C[9]).

The RDF data model [KC04] allows structured representations of web resources to be made, by expressing *statements* about web resources in the form of *triples*.

**Definition 6.** We consider $U$ as being the set of all web identifiers, $L$ as being the set of all literals and $B$ the set of all *blank nodes* (corresponding to existentially quantified variables, as defined in Hayes's RDF semantics [Hay04]). $U$, $L$ and $B$ are pairwise disjoint. Let $N = U \cup L \cup B$, then the set $T = N \times U \times N$ is the set of all RDF *triples* (or *statements*)[10]. We call the first element of a RDF triple the *subject*, the second element the *property* and the third the *object*.

An example RDF triple is depicted in Figure 2.1. RDF triples correspond to binary predicates in FOL. For example, the triple in Figure 2.1 corresponds to the following predicate.

$$\text{‘http://xmlns.com/foaf/0.1/name’}(\text{‘http://dbtune.org/jamendo/artist/5’}, \text{“Both”})$$

[8]`http://www.w3.org/PICS/`

[9]`http://www.w3.org/`

[10]The RDF specification holds a legacy constraint that a literal cannot be the subject of an RDF triple. However, this will probably change, as noted by the RDF-core working group at `http://www.w3.org/2000/03/rdf-tracking/#rdfms-literalsubjects`. In all this thesis, we consider that a literal can be the subject of an RDF triple.

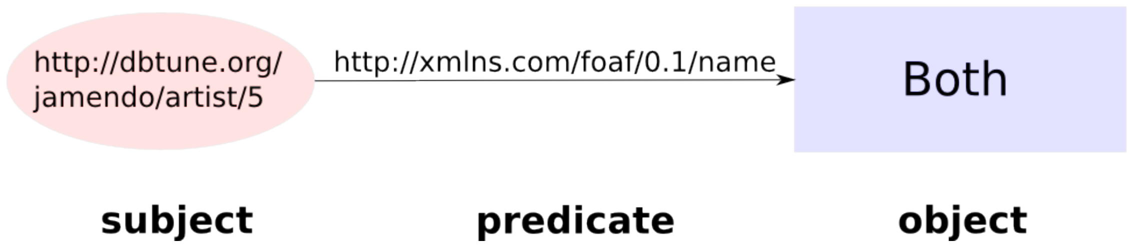


Figure 2.1: A RDF triple representing the piece of information that the band identified by `http://dbtune.org/jamendo/artist/5` has the name "Both"

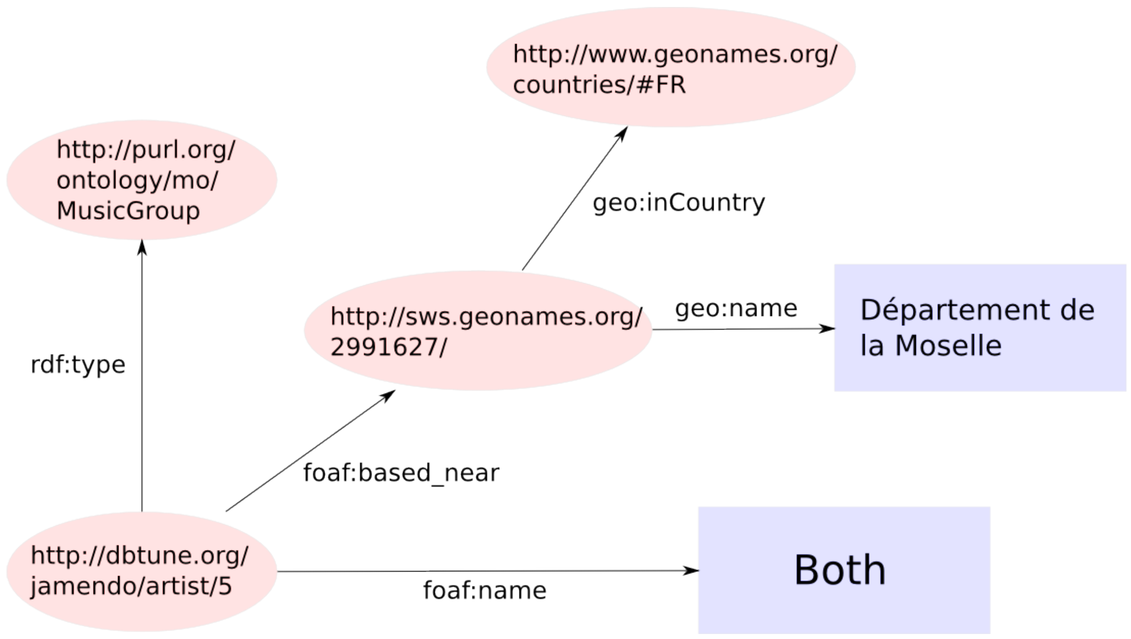


Figure 2.2: A RDF graph representing that a band has a name "Both" and is based near the 'Département de la Moselle', in France

A set of triples may be interpreted as a graph of these resources, with arcs corresponding to the relationships between them. An example of such a graph is depicted in Figure 2.2. The scope of the existential quantification is the graph in which the existentially quantified variable occurs [Hay04].

The RDF model includes a *reification* mechanism. A RDF triple can be explicitly identified by a web identifier. This web identifier can be used to state things about that particular statement (e.g. "this person made that statement"). However, it has been suggested that reifying RDF information at the triple level is not the best approach [Biz07] for representing such meta-information. Another approach is to extend the RDF model with a support for Named Graphs [CBHS05b]. In the Named Graphs approach, we consider reifying a whole RDF graph, and give it a URI. This URI can then be further described in other RDF graphs.

**Definition 7.** The set of RDF graphs $G$ is the power set of $T$, as defined in Definition 6. A named graph is then a pair $ng = (u, g)$, where $u \in U$ and $g \in G$.

In an aggregation of structured web data, the graph URI $u$ may correspond to the source of the statements. We can attach to the graph URI information about the last accessed time, the time required to access and parse the corresponding document, etc.

#### 2.2.4.3 RDF syntaxes

Several syntaxes exist for RDF.

The RDF/XML syntax[11] is an XML-based syntax for RDF. The RDF representation of the URI `http://dbtune.org/jamendo/artist/5`, requested using the HTTP Accept header, is expressed in RDF/XML as follows.

```
<rdf:RDF
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
  xmlns:mo="http://purl.org/ontology/mo/"
  xmlns:owl="http://www.w3.org/2002/07/owl#">
<mo:MusicGroup rdf:about="http://dbtune.org/jamendo/artist/5">
  <foaf:made rdf:resource="http://dbtune.org/jamendo/record/174"/>
  <foaf:made rdf:resource="http://dbtune.org/jamendo/record/33"/>
  <owl:sameAs
 rdf:resource="http://dbtune.org/musicbrainz/resource/artist/
0781a3f3-645c-45d1-a84f-76b4e4decf6d"/>
  <foaf:based_near rdf:resource="http://sws.geonames.org/2991627/"/>
  <foaf:homepage rdf:resource="http://www.both-world.com"/>
  <foaf:img rdf:resource="http://img.jamendo.com/artists/b/both.jpg"/>
  <foaf:name rdf:datatype="&xsd;string">Both</foaf:name>
</mo:MusicGroup>
</rdf:RDF>
```

This resource is a band (where the 'band' concept is captured by `mo:MusicGroup`, as explained in § 2.2.6), it made two albums, it is the same as another web resource, it has a homepage, an image and a name. We note that several other web identifiers are quoted within this representation, allowing us to discover more structured data. For example, we can follow the `foaf:based_near` link to a resource in the Geonames dataset[12] in order to get detailed information about the place where this band is based.

In this thesis, we use the Turtle RDF syntax[13], a sub-set of the N3 syntax we describe in § 2.2.4.5, for our RDF examples. The above RDF example is expressed in Turtle as follows.

```
<http://dbtune.org/jamendo/artist/5>
  a mo:MusicGroup;
  foaf:made <http://dbtune.org/jamendo/record/174>;
  foaf:made <http://dbtune.org/jamendo/record/33>;
  owl:sameAs <http://dbtune.org/musicbrainz/resource/artist/
0781a3f3-645c-45d1-a84f-76b4e4decf6d>;
  foaf:based_near <http://sws.geonames.org/2991627/>;
  foaf:homepage <http://www.both-world.com>;
  foaf:img <http://img.jamendo.com/artists/b/both.jpg>;
  foaf:name "Both"^^xsd:string.
```

[11] `http://www.w3.org/TR/rdf-syntax-grammar/`
[12] `http://geonames.org/`.
[13] `http://www.w3.org/TeamSubmission/turtle/`

Each block corresponds to a set of statements (subject, predicate, object) about one subject. web identifiers are either between angle brackets or in a `prefix:name` notation, with the namespaces defined in Appendix E. Literals are enclosed in double quotes and can hold an explicit typing information using `^^`. The keyword `a` corresponds to the Web identifier `rdf:type`. Blank nodes are prefixed by an underscore, or are denoted by square brackets holding a set of statements about them. For example:

```
<http://dbtune.org/jamendo/artist/5>
  foaf:maker [
    a mo:Track ;
    dc:title ‘‘Une charogne" ].
```

and

```
<http://dbtune.org/jamendo/artist/5>
  foaf:maker _:track .
_:track
  a mo:Track ;
  dc:title ‘‘Une charogne" .
```

Both encode to the same RDF: “the artist identified by `http://dbtune.org/jamendo/artist/5` made a track whose title is “Une charogne” ”.

#### 2.2.4.4 Microformats and GRDDL

Another way to publish RDF data on the Web is to use Microformats[14] or RDFa[15].

Microformats were designed to embed structured data in XHTML[16] documents. For example, using the hAudio microformat, we can express that an XHTML document holds information about a track and its creator:

```
<div class="haudio">
  <span class="title">Une charogne</span> by
  <span class="contributor">Both</span>
</div>
```

A web browser would just show the text `Une charogne by Both`, but a Microformats-aware browser can use the structured data for other purposes (aggregating browsed artists and tracks, for example). However, Microformats are less expressive than RDF, as they cover a limited set of domains instead of providing a generic data model.

Recent efforts in embedding full RDF annotations in XHTML web documents led to the development of RDFa. For example, the same annotation as above (a track and its creator) would be written in RDFa as follows.

```
<div typeof="mo:Track" about="#track"
  xmlns:foaf="http://xmlns.com/foaf/0.1/"
```

[14]http://microformats.org/
[15]http://rdfa.info/
[16]http://www.w3.org/TR/xhtml11

```
  xmlns:mo="http://purl.org/ontology/mo/"
  xmlns:dc="http://purl.org/dc/elements/1.1/" >
  <span property="dc:title">Une charogne</span> by
  <div rel="foaf:maker">
    <div typeof="mo:MusicGroup" about="#artist">
      <span property="foaf:name">Both</span>
    </div>
  </div>
</div>
```

This gives the following RDF, in Turtle syntax:

```
<#track>
  a mo:Track ;
  dc:title "Une charogne" ;
  foaf:maker <#artist> .
<#artist>
  a mo:MusicGroup ;
  foaf:name "Both" .
```

The track is identified by `<#track>`, the artist is identified by `<#artist>`, and we refer to external identifiers, such as `foaf:maker` or `mo:Track`.

The conversion step from Microformats or RDFa to RDF uses the GRDDL specification[17]. A web document can link to a document specifying a transformation. This transformation allows automated agents to extract RDF data.

#### 2.2.4.5 N3 data model and syntax

N3 [BLCK+08] is both a super-set of the RDF data model and an associated syntax. The Turtle syntax described in § 2.2.4.3 is a subset of this syntax. The data model encompasses *variable nodes* (universally quantified variables).

**Definition 8.** We consider $U$ as being the set of all web identifiers, $L$ as being the set of all literals, $B$ as being the set of all blank nodes and $V$ as being the set of all variable nodes. $U$, $L$, $B$ and $V$ are pairwise disjoint. Let $N = U \cup L \cup B \cup V$ and $UV = U \cup V$, then the set $T = N \times UV \times N$ is the set of all N3 statements.

In the corresponding syntax, variable nodes start with `?`. If a single statement involves both universal and existential quantification, then the scope of the universal quantification is outside the scope of the existential quantification.

N3 also encompasses quoting. A literal resource can correspond to a set of statements (a graph).

**Definition 9.** We consider $G$ as being the power set of $T$. Then, $G \subset L$.

We refer to such resources as *graph literals*. In the corresponding syntax, graph literals are denoted by curly brackets.

[17] `http://www.w3.org/TR/grddl/`

Using quoting and variable nodes, N3 can be used to represent FOL sentences. A graph literal corresponds to the conjunction of all the statements it holds, and the `log:implies` property corresponds to $\supset$. The keyword `=>` in N3 is a shortcut for `log:implies`, and the keyword `<=` for the inverse property of `log:implies`. For example,

```
{?a a mo:Conductor} => {_:p mo:conductor ?a; a mo:Performance}.
```

corresponds to

$$\forall A.\ \mathsf{mo\!:\!Conductor}(A) \supset \exists P.\ \mathsf{mo\!:\!Performance}(P) \wedge \mathsf{mo\!:\!conductor}(P, A)$$

Quoting also provides a form of reification, similar to the Named Graphs mechanism mentioned in § 2.2.4.2. The main difference lies in the fact that the graphs does not need to be explicitly named. The identity of a graph $g \in G$ is its value and only its value. Further information can be attached to a graph literal, as follows.

```
{ :performance mo:conductor [ foaf:name "Robert Shaw" ] } :believedBy :john .
```

The fact that the performance `:performance` involved a conductor named Robert Shaw is believed by the person `:john`.

### 2.2.5 Semantic Web user agents

We use the term “user agent” to describe any software acting directly on user requests, and a “Semantic Web user agent” is then one which accesses resources on the Semantic Web in order to satisfy a user’s demands. Semantic Web user agents can possibly perform information integration, reasoning and deduction on behalf of a human user. One example of a Semantic Web user agent would be a simple browser, analogous to a modern web browser, which allows the user to navigate data on the Semantic Web just as a web browser allows a user to navigate web sites. The Semantic Web is designed from the outset to allow much more complex interaction with available data sources, and so the term “Semantic Web user agent” also encompasses more complex modes of data consumption.

This includes programs which automatically explore extra resources in order to satisfy a user’s query. A simple example is the Tabulator generic data browser [BLCC$^+$06] which automatically accesses resources if they are semantically marked “the same as” or “see also” from a retrieved resource, and then provides the user with a conflated view of all the information found. More complex examples include the ‘Semantic Web Client Library’ and our SWIC software[18], which aggregate structured web data in order to satisfy an arbitrary query.

Semantic Web user agents may also express complex queries directly to remote RDF data publishers, for example using the SPARQL[19] query language to submit the query to a publisher’s *SPARQL endpoint*. The query language allows requests ranging from simple DESCRIBEs (“return all information about resource $x$”) to complex structured queries about the endpoint’s database (e.g. “return the latest album from each artist who had an album in the US charts in the 70s”). SPARQL also handles Named Graphs, as defined in § 2.2.4.2. RDF data can be queried within a particular graph, identified by a URI.

[18]see `http://www4.wiwiss.fu-berlin.de/bizer/ng4j/semwebclient/` and `http://moustaki.org/swic/`
[19]see `http://www.w3.org/TR/rdf-sparql-query`

### 2.2.6 Web ontologies

Web identifiers can also be used to denote concepts (e.g. the `mo:MusicGroup` URI in the first example of § 2.2.4.3), roles (e.g. the `foaf:maker` URI in the same example) and individuals (e.g. the group URI), as defined in § 2.1.6. A web ontology is a set of web identifiers and corresponding structured representations specifying the important concepts and relationships in a given domain. For example, a user agent can follow the link to the `mo:MusicGroup` URI and see where this term fits within a conceptual framework, how it relates to other concepts, what roles can be applied to it, etc.

Several web-based ontology languages exist for such representations.

#### 2.2.6.1 SHOE

An early web ontology language is the SHOE language [Hef01], which is not RDF-based. SHOE ontologies are described in a web document, which specifies different categories (unary predicates, as defined in § 2.1.2) and relationships (n-ary predicates). However, the amount of currently available SHOE data is limited. Its RDF-based counterparts, which we study in the following, are more widely used.

#### 2.2.6.2 RDF Schema

The RDF vocabulary definition language RDF Schema [BG04] provides a simple web ontology language. RDF Schema can be used to define classes, properties, hierarchies of classes and properties, and domain and range constraints for properties.

For example, a simple RDF Schema ontology is as follows.

```
mo:Record
  a rdfs:Class ;
  rdfs:label "Record" .
mo:Track
  a rdfs:Class ;
  rdfs:label "Track" .
mo:track
  a rdf:Property ;
  rdfs:label "track";
  rdfs:domain mo:Record;
  rdfs:range mo:Track .
```

We here define the class of all records and the class of all tracks, and we define a property to link a record to a track.

#### 2.2.6.3 OWL

The expressiveness of RDF Schema is limited. For example, we cannot express that all persons conducting a performance are conductors, as in § 2.1.2.1. This issue led to the specification of the Web Ontology Language (OWL [MvH04]). OWL extends RDF schema with new modelling primitives, for example to define properties that are inverses of each other, properties that are

| Constructor | OWL-DL term |
|---|---|
| Concept | `owl:Class` |
| Role | `owl:ObjectProperty` |
| Top | `owl:Thing` |
| Bottom | `owl:Nothing` |
| Conjunction | `owl:intersectionOf` |
| Disjunction | `owl:unionOf` |
| Complement | `owl:complementOf` |
| Restriction | `owl:Restriction` + `owl:onProperty` |
| Universal Restriction | `owl:allValuesFrom` |
| Existential Restriction | `owl:someValuesFrom` |
| Subsumption | `rdfs:subClassOf` |
| Role hierarchy | `rdfs:subPropertyOf` |
| Inverse role | `owl:inverseOf` |
| Transitive role | `owl:TransitiveProperty` |
| Functional role | `owl:FunctionalProperty` |
| Collection of individuals | `owl:oneOf` |
| Classification | `rdf:type` |

Table 2.2: $\mathcal{SHOIN}(D)$ constructors mentioned in § 2.1.6 and corresponding terms in OWL-DL

transitive, properties that are functional, value constraints and restrictions. OWL is based on the Description Logics reviewed in § 2.1.6 [BHS05].

OWL has three different sub-languages, defined by the modelling primitives they support, with different formal complexities and expressiveness:

- OWL-Lite – This sub-language has the lowest formal complexity in the OWL family;
- OWL-DL – This sub-language has the best expressiveness whilst still being decidable and corresponds to the $\mathcal{SHOIN}(D)$ Description Logic described in § 2.1.6. All the constructors in table 2.1 are expressible within OWL-DL, as shown in table 2.2. We use OWL-DL and the $\mathcal{SHOIN}(D)$ Description Logic for specifying ontologies in this thesis;
- OWL-Full – This sub-language includes all the modelling primitives available in OWL, and is therefore the most expressive. However, it does not offer any computational guarantee.

## 2.3 Summary

We reviewed the areas which constitute the foundations of our research: Knowledge Representation and Semantic Web technologies. Description Logics and FOL, described in § 2.1, are the foundations of our knowledge representation frameworks, which we specify in chapters 3 and 5. The web technologies described in § 2.2 are used throughout this thesis as a foundation for the distributed aspects of our system.

Part II

# A web-based music information system

# Chapter 3

# Conceptualisation of music-related information

In order to use the Web for publishing music-related information, we need to create a web ontology for the music domain, as described in § 2.2.6. Such an ontology would provide a set of identifiers and corresponding descriptions for concepts and relationships in the music domain, that music-related data can link to. Then, a user agent wanting to know how to process a particular music-related resource can follow a link to these ontological terms and be provided with their descriptions. This is the purpose of our *Music Ontology* framework [RASG07, AR06], providing web identifiers and associated structured representations for an ontology of the music domain.

After giving a review of previous representation systems in the music domain in § 3.1, we define in § 3.2 and § 3.3 a set of terms (concepts and relationships) that cover a wide range of music-related information. We will evaluate how wide this range is in chapter 4. We discuss this ontology of the music domain in § 3.4.

## 3.1 Models of the music domain

Machine-processable representations of information in the music domain are a necessary step towards truly integrated computer music environments that are able to help retrieving music-related information, to assist the formulation of theories about musical understanding and test them. A music representation system then provides a basis for music information retrieval and automating systematic musicology. For these reasons, several music representation frameworks [Dan93] have been created. We classify these frameworks in three different categories, based on the form of their underlying data model. We focus on frameworks that are based a layered data model in § 3.1.1, on frameworks that are based on a hierarchical data model in § 3.1.2 and on workflow-based frameworks in § 3.1.3.

### 3.1.1 Layered modelling of the music domain

As Dannenberg points out [Dan93], musicians have to deal with many layers of abstractions: from the music structure to the emotion a particular performance might trigger, for example. Several music representation frameworks characterise such a layering in abstraction, from the most abstract musical entity to the most concrete. We here review some of these frameworks.

| Example 1 | "Trout Quintet" |
|---|---|
| Work | Franz Schubert's Trout Quintet |
| Expression | The composer's score<br>Sound issued from a performance by the Cleveland Quartet |
| Manifestation | Recording of the Cleveland Quartet, released on Compact Disc |
| Example 2 | "Love Buzz" |
| Work | Shocking Blue's Love Buzz (composed in 1969) |
| Expression | Love Buzz, as performed by Shocking Blue in 1969<br>Love Buzz, as performed by Nirvana in 1990 |
| Manifestation | Track in the Nirvana "Bleach" album released in June 1989 |

Table 3.1: Example of FRBR instances

#### 3.1.1.1 Conceptual layering in the music domain

The HARP system [CCIM95] is built around two main abstraction layers. A *symbolic* layer deals with abstract representations and reasoning mechanisms. This symbolic layer covers the set of rules and symbols used to represent a musical work. A *sub-symbolic* layer deals with dense data, e.g. signal samples.

Vinet [Vin04] defines four main abstraction layers for musical information: *physical*, e.g. physical sounds, *signal*, e.g. digital, analogue, encoded or compressed representation of a physical sound, *symbolic*, i.e. discrete representation based on an enumerable set of symbols and *knowledge*, e.g. higher-level specification of symbolic information.

The MX music representation framework [HL05a] defines six abstraction layers: *audio*, *performance*, *notational*, *logical*, *structural* and *general*. MX leaves Vinet's physical layer out, but adds several layers between symbolic information and an abstract entity corresponding to the musical work itself.

A layering in abstraction is necessary when dealing with music-related information. A requirement for our framework is therefore to provide such a layering.

#### 3.1.1.2 FRBR

The Functional Requirements for Bibliographic Records (FRBR [otFRfBR98, DN05]) is an example of an ontology specifying a layered representation system. Four distinct levels of abstraction are considered:

- *Work* – A distinct intellectual creation;
- *Expression* – An artistic realisation, for example the product of a musical performance;
- *Manifestation* – A group of similar items embodying an expression, for example an album;
- *Item* – A single exemplar of a manifestation, for example an audio file or a copy of an album.

Table 3.1 gives some example instances of these concepts.

FRBR has been successfully applied in a digital music library context [RHCB07], and we consider reusing this layering in abstraction within our framework. The relationships between FRBR's, Vinet's and MX's layering can be depicted as in Figure 3.1.

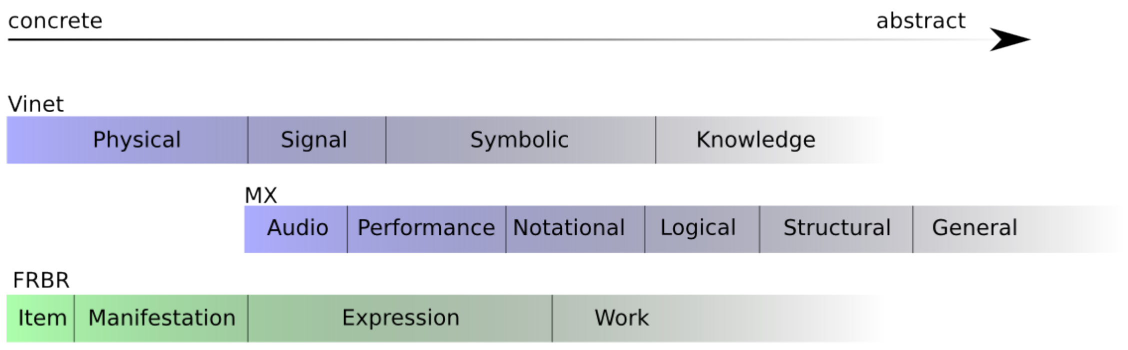


Figure 3.1: Relationships between abstraction layers in Vinet's work, MX and FRBR

### 3.1.2 Hierarchical modelling of the music domain

Another set of music representation systems develop a hierarchical view of musical information. Their common feature is to take a particular range of abstractions as a starting point, e.g. digital items, performance or score information, and to develop a hierarchical representation of it.

#### 3.1.2.1 Performances and scores

Certainly, the most well-known hierarchical representation system is the system that Schenkerian analysis [Sch04] leads to. Starting from a musical work (the *foreground*), we get to its underlying structure (the *fundamental structure*) in several nested layers, as formalised by Regener [Reg67]. Buxton et al. [BRBM78] describe a hierarchical symbolic representation system, in which a composition is expressed as a sequence of events, where events can be a musical note or a sequence of events. A similar representation system is the XML-based Music Encoding Initiative [Rol02]. Other systems consider these events as reified entities, therefore allowing cross-references of events and reuse of compound objects. The CHARM system [HSW91, SWH93] describes such a hierarchical representation system. CHARM can be used to specify performance information. A musical score that does not determine the order of some musical events corresponds to several CHARM representations. Musical information is clustered into nested constituents and sub-constituents, down to the level of individual events.

Balaban [Bal96] describes a music representation system based around the notion of *structured music pieces* (SMPs), capturing time and hierarchy information and a set of arbitrary information (*methods*). An important aspect of this representation framework is that musical information can be under-specified. For example, timing information can be incomplete. A musical event can be characterised as occurring before another event within a SMP, without being explicit on its start and end time. Such under-determination allows us to tackle a wider range of abstraction layers than in the CHARM system.

Balaban's framework shows that a hierarchical model on top of an appropriate time model can be very expressive. We can detail part–whole relationships at different levels of abstractions, for example at the score level and at the performance level. A requirement for our representation framework is therefore to develop a sufficiently expressive temporal model, supporting hierarchical representations of different abstractions layers.

#### 3.1.2.2 Audio items

Pachet [Pac05] develops a hierarchical representation system, whose root node corresponds to musical audio items. He distinguishes three clusters of information related to audio items.

- *Editorial* – Information pertaining to the creation of the item, such as the performers and the musical work performed;
- *Cultural* – Inferred from an analysis of emerging patterns, such as the results derived from an analysis of listening habits data;
- *Acoustic* – Representations derived from the content of the item.

In general, 'meta-data' frameworks for audio items, such as ID3 [id3] for MP3 audio files or M3F [m3f] for OGG audio files, lead to similar representation systems. They allow textual information to be attached to the items. Their main goal is to help manage collections of audio files, by providing a way to locate content within them using this textual information, e.g. artist name, track name, album name, average number of beats per minute.

However, these framework usually limit themselves to one level of description. They do not allow a resource else than the audio item to be described further, for example an artist by his date of birth, his country, his discography, etc. Our framework needs to consider more resources than just the audio items, such that all resources can provide an anchor point for more information.

#### 3.1.2.3 Dublin Core

A typical example of a Semantic Web meta-data framework is the Dublin Core [dubb] Metadata Initiative (DCMI). Dublin Core is an attempt at providing a set of shared terms for describing multimedia items, that can be subsumed in specialised vocabularies. The 15 Dublin Core terms are represented in table 3.2, where they are clustered in three categories. Dublin Core terms are related to the content of the multimedia item, related to the item itself, or related to the publication of the item. These terms are published in the form of a RDF vocabulary [duba], and give the possibility of attaching qualified textual descriptions to web resources. An example of such a description is the following, describing an audio item:

```
:axel-both
   dc:date "2008-02-19";
   dc:creator "Both";
   dc:title "Axel" .
```

We consider reusing some Dublin Core terms within our framework in a context where the items to annotate have a relatively low degree of abstraction. This includes items related to the publication of an audio signal (such as albums or audio files), edited musical scores, etc. For example, we would use `dc:title` to attach an album to a string corresponding to its title.

#### 3.1.2.4 MPEG-7

MPEG-7 [NL99a, NL99b] provides an XML-based framework for describing audio-visual resources. MPEG-7 was designed to make audio-visual items available on the Web as searchable as text. The associated model is hierarchical, starting at a really concrete level, the audio-visual

| Content | Item | Publication |
|---|---|---|
| Title | Format | Creator |
| Subject | Date | Publisher |
| Description | Language | Contributor |
| Type | Identifier | Rights |
| Source | | |
| Relation | | |
| Coverage | | |

Table 3.2: Dublin Core elements

item itself, to derive a number of abstractions (*descriptors* or *features*) based on the content, as described in Appendix A. For example, MPEG-7 allows MFCCs, as described in § $A.4$, to be attached to an audio item. Extra information can also be added, such as a title or a description. MPEG-7 has a number of limitations [vONH05, TC04, CDH$^+$07, Tro03]. The following aspects of MPEG-7 make us move away from using it as a basis for our framework.

- The difficulty of mixing vocabularies from heterogeneous domains. MPEG-7 tends to limit the expression of multimedia-related information by explicitly restricting the kind of information MPEG-7 documents may contain. While this approach can work very well for closed, isolated data sets, it is inappropriate for building a large-scale information environment, where such enforced domain boundaries are artificial and should be avoided if at all possible. We detail this point in § 3.4.2.
- The lack of formal semantics in MPEG-7 makes it difficult to perform automated reasoning tasks on top of MPEG-7 information.
- The inability to use URIs as identifiers within MPEG-7 descriptions makes it difficult to distribute and interlink data across multiple locations on the Web. For example, when annotating a particular recording as involving Glenn Gould, we want to simply refer to him by a web identifier[1] giving access to more information instead of having to redescribe him.
- The lack of an expressive conceptual model for complex music-related information e.g. performances, arrangements, recordings or musical works.

### 3.1.3 Workflow-based modelling

Instead of defining a minimal set of properties characterising web resources (such as Dublin Core) or a minimal set of abstraction layers (such as FRBR), the ABC ontology described by Lagoze et. al in [LH01] tries to encompass the most information possible in a cross-domain digital library context. ABC provides a meta-modelling facility, allowing it to coordinate heterogeneous representation models. This is illustrated in [Hun03], where Hunter describes how MPEG-7 and MPEG-21 can be integrated in the ABC framework. The ABC ontology is centred around the notion of *situations* and *events*, where an event is considered as the transition from a situation to another. An interesting feature of ABC is the inclusion of resources that do not exist within the context of a particular situation. These abstract resources, e.g. a musical work, allow us to tie up several manifestations of it, e.g. performances or scores, occurring in the same or in different *workflows* i.e. sets of situations and corresponding events. ABC therefore provides two

[1] such as his DBpedia URI `http://dbpedia.org/resource/Glenn_Gould`

levels of abstractions loosely corresponding to manifestations and works in FRBR terminology and the ability to specify any type of workflow linking different manifestations together. Such an event-based integration methodology is also advocated by Ruotsalo and Hyvonen [RH07] and in the CIDOC-CRM model [Doe03].

Although we will argue some aspects of the ABC model § 3.2.2, the modelling we adopt for music production workflows is based on similar concepts. We consider multiple types of abstractions such as scores, sounds or signals, and their complex inter-relationships are modelled as chains of events.

## 3.2 Core elements of a Music Ontology

In the following sections, we describe our Music Ontology framework. This framework consists of a set of Semantic Web ontologies for dealing with a wide range of music-related information, from detailed editorial and production information to temporal annotations.

### 3.2.1 Time

Temporal information is the first thing we want to express when dealing with music-related information. Indeed we need to cover a large range of temporal descriptions, from 'this performance happened on the 9th of March, 1984' to 'this beat is occurring around sample 32480', to 'the first chorus of this song is just before the second verse'. We design an ontology able to express such temporal information: the Timeline Ontology [RA06b].

We here give a brief introduction of different temporal theories and associated ontologies (we refer the interested reader to Patrick Hayes's "Catalog of Temporal Theories" [Hay96] for a more detailed review), and then describe our Timeline Ontology.

#### 3.2.1.1 Overview of temporal theories

We here only consider the actual structure of time, instead of the way time triggers *changes*. Time is a physical dimension as defined by Gruber & Olsen [GO94]. The idea of time is associated with a temporal continuum, which may be considered as linear (in this case, we refer to it as a 'timeline') or branching (we then refer to the more general notion of a 'time-plenum'). For example, planning systems, trying to find a sequence of actions that would satisfy a particular goal, consider several possible *futures*, as in the temporal logic defined by McDermott [McD82]. Time points and time intervals are objects whose sole dimension is time. Time intervals can have an extent, called a duration. Temporal coordinate systems are used to position temporal entities. For example, "midnight, the 9th of March, 1984" uses a calendar-based coordinate system, but refer to the same temporal entity as 447638400 in Unix time[2].

We therefore consider six different time-related concepts: dimension, plenum, interval, duration, point and coordinate system. All temporal theories play with these six concepts, but the relationships between them can be conceptualised in a number of ways. We may consider points as first-class objects. An interval is then identified by the points it contains. Or we may consider intervals as first-class objects. Points then become places where intervals meet or zero-width intervals. Allen [All83] defines a number of interval-to-interval relationships and a calculus to

[2]number of seconds since midnight UTC, January 1st, 1970

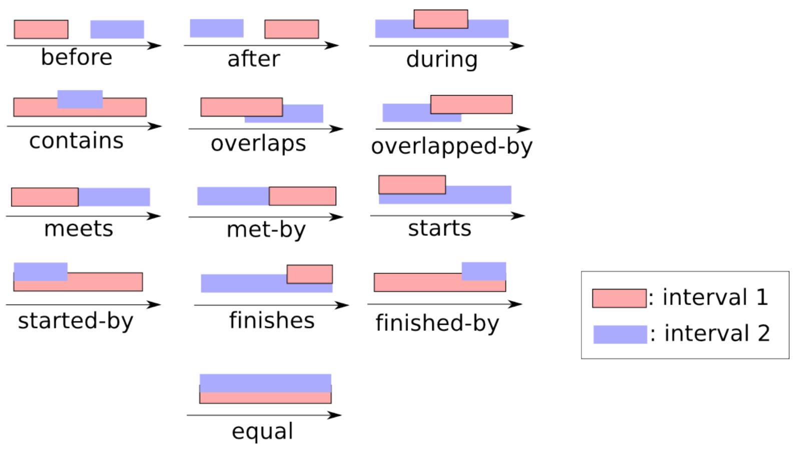


Figure 3.2: Allen's 13 interval-to-interval relationships. Arrows represent time. Interval 1 and interval 2 are in the relationship specified under the arrow.

keep track of interval-related information, as depicted in Figure 3.2. Allen leaves the notion of time points aside, as it leads to counter-intuitive statements, although Hayes [Hay96] conciliates to some extent these two views.

The HyTime ISO standard [Gol91] defines a time model to address and relate temporal fragment of different time-based multimedia documents. It extends the model used in another SGML-based ISO standard, the Standard Music Description Language (SMDL [Slo97]). Hence, it is based around the concept of an 'abstract time' and a 'real time', in which particular fragments ('events') can be projected onto one another. Our Timeline and Event ontologies described in § 3.2.1.2 and § 3.2.2 generalise this model.

#### 3.2.1.2 The Timeline ontology

The OWL-Time ontology [HP04, Pan07] formalises these different notions in a Semantic Web framework, providing web identifiers for a number of concepts we mentioned earlier: instant, interval, Allen's relationships, and coordinates in the Gregorian calendar.

However, the OWL-Time ontology is not directly usable for all music-related information. We indeed need to consider the existence of multiple timelines, and the relationships between them. For example, we might want to express a time interval on a timeline corresponding to a particular digital signal (e.g. "from sample 13345 to sample 32578 on this particular signal"), and we may want to relate such an interval with the corresponding interval on the physical timeline (e.g. "this particular interval corresponds to 0.43 seconds of a recording of this performance, which started at 08:53 on 12/03/08").

We hence extend OWL-Time and introduce a new timeline concept, depicted in Figure 3.3, and defined as follows:[3].

$$\mathsf{tl{:}TimeLine}(L) \supset \mathsf{owl{:}Thing}(L) \tag{3.1}$$

[3]In the following, we use a FOL syntax for specifying our ontologies, as described in § 2.1.2. All variables start with an upper case character. All variables in the antecedent of the implication are universally quantified. Unary predicates correspond to concepts, and binary predicates to roles. All the terms are prefixed by a namespace. These namespaces are defined in Appendix E. The mapping to the corresponding DL syntax is given in table 2.1. The mapping to OWL-DL/RDF is given in table 2.2. This syntax makes it easier to identify how the different ontology axioms can be implemented as a logic program [GHVD03] or as non-updating rules within our N3-Tr knowledge representation framework described in chapter 5.

where the owl:Thing concept is interpreted as the set of all individuals, as defined in § 2.1.6.

A timeline supports time instants and intervals through the following predicate:

$$\mathsf{tl{:}timeline}(T, L) \supset \mathsf{time{:}TemporalEntity}(T) \wedge \mathsf{tl{:}TimeLine}(L) \tag{3.2}$$

where $\mathsf{time{:}TemporalEntity}(T) \supset \mathsf{tl{:}Interval}(T) \vee \mathsf{tl{:}Instant}(T)$.

A single timeline may be associated to several temporal coordinate systems. These coordinate systems can be used to address time points and intervals on the corresponding timeline. We subsume the timeline concept to handle continuous, discrete and *abstract* timelines. For example, a digital signal is supported by a discrete timeline, whereas an analogue signal is supported by a continuous timeline. An entity that does not need to be positioned throught the use of a temporal coordinate system is associated to an abstract timeline. For example, the ordering of the movements in a work can be expressed by ways of relationships among them, e.g. before and after.

$$\mathsf{tl{:}ContinuousTimeLine}(L) \vee \mathsf{tl{:}DiscreteTimeLine}(L) \vee \mathsf{tl{:}AbstractTimeLine}(L) \supset \mathsf{tl{:}TimeLine}(L) \tag{3.3}$$

A continuous timeline can either be relative, e.g. the timeline supporting a recorded analogue signal, or physical. An instance of a physical timeline is the *universal* timeline, supporting for example calendar information.

$$\mathsf{tl{:}ContinuousTimeLine}(L) \equiv \mathsf{tl{:}PhysicalTimeLine}(L) \vee \mathsf{tl{:}RelativeTimeLine}(L) \tag{3.4}$$

$$\mathsf{tl{:}PhysicalTimeLine}(\mathsf{tl{:}universaltimeline}) \tag{3.5}$$

Our ontology also defines a way to relate two timelines together and to reify [BB03] this relationship, through the definition of a 'timeline map' concept.

$$\mathsf{tl{:}TimeLineMap}(M) \supset \mathsf{owl{:}Thing}(M) \tag{3.6}$$

$$\mathsf{tl{:}domainTimeLine}(M, L) \vee \mathsf{tl{:}rangeTimeLine}(M, L) \supset \mathsf{tl{:}TimeLineMap}(M) \wedge \mathsf{tl{:}TimeLine}(L) \tag{3.7}$$

Instances of this concept can be used, for example, to express the relationship between the continuous timeline supporting an analogue signal and the discrete timeline supporting the sampled signal. Or, by using the abstract timeline defined above, to link parts of a musical work to parts of a given performance. Such maps are captured through sub-concepts of the timeline map concept. For example, the origin-map concept provides a way to relate a relative timeline to the physical timeline. It can be used to express the relationship between the timeline supporting a recorded analogue signal and the timing of the corresponding performance. The uniform sampling map concept provides a way to relate the timeline of an analogue signal to the timeline of the corresponding digitised signal. These different kinds of timeline maps are defined as follows.

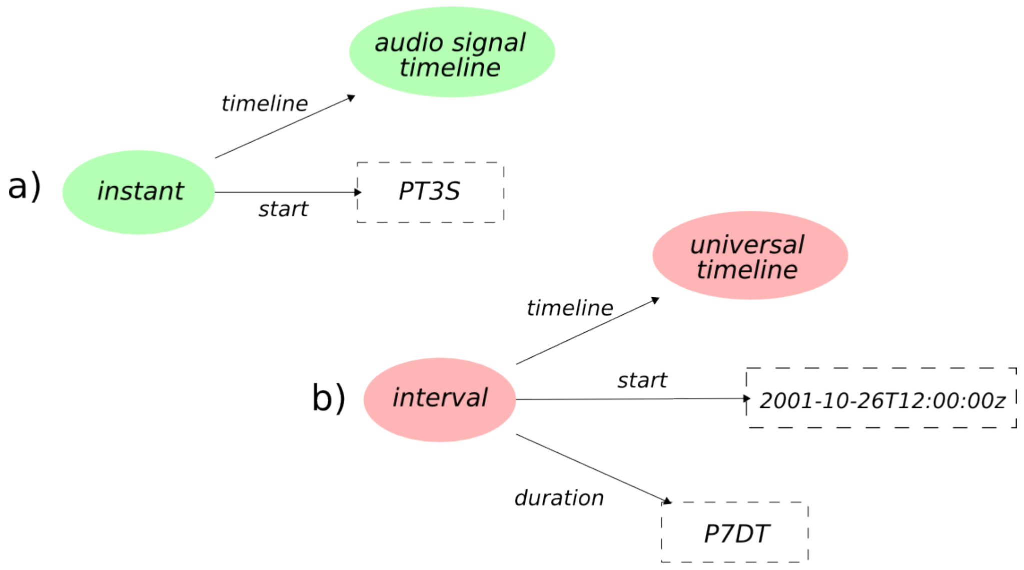


Figure 3.3: (a) Describing an instant on an audio signal timeline (at 3 seconds). (b) Describing an interval on the universal timeline (7 days starting on 26 October 2001, 12:00 UTC)

$$\text{tl:OriginMap}(M) \vee \text{tl:UniformSamplingMap}(M) \vee \text{tl:UniformWindowingMap}(M) \supset \text{tl:TimeLineMap}(M) \tag{3.8}$$

$$\text{tl:domainTimeLine}(M, L) \wedge \text{tl:OriginMap}(M) \supset \text{tl:PhysicalTimeLine}(L) \tag{3.9}$$

$$\text{tl:rangeTimeLine}(M, L) \wedge \text{tl:OriginMap}(M) \supset \text{tl:RelativeTimeLine}(L) \tag{3.10}$$

$$\text{tl:domainTimeLine}(M, L) \wedge \text{tl:UniformSamplingMap}(M) \supset \text{tl:RelativeTimeLine}(L) \tag{3.11}$$

$$\text{tl:rangeTimeLine}(M, L) \wedge \text{tl:UniformSamplingMap}(M) \supset \text{tl:DiscreteTimeLine}(L) \tag{3.12}$$

$$\text{tl:domainTimeLine}(M, L) \wedge \text{tl:UniformWindowingMap}(M) \supset \text{tl:DiscreteTimeLine}(L) \tag{3.13}$$

$$\text{tl:rangeTimeLine}(M, L) \wedge \text{tl:UniformWindowingMap}(M) \supset \text{tl:DiscreteTimeLine}(L) \tag{3.14}$$

(3.15)

This ontology is primarily available in the OWL-DL sub-language of the Ontology Web Language mentioned in § 2.2.6.3, and published at [RA06b]. An example of two temporal entities on two different timelines is depicted in Figure 3.3.

### 3.2.2 Event

Now, we consider *classifying* regions of such timelines, to express things such as "the first chorus of this song is here" or "that musical work was performed at that time". We design an ontology able to express such classifications: the Event Ontology [RA06a, ARS06]. We first give a review of related event modelling approaches, and then describe ours.

#### 3.2.2.1 Event modelling

The ontological status of an event in the ABC ontology described in § 3.1.3 is defined as being the transition from one situation to another. For example, the lending of a painting from one museum to another can be modelled as a transition from a situation in which the painting is located in the first museum to a situation in which the painting is located in the second museum. A similar definition of an event is given by Francois et al. [FNH+05] in a video annotation context: "A change of state in an object triggers an event".

The event calculus[4] [KS86] models event types as implying whether situational fluents ('being alive', for example) holds or stops holding. It therefore proposes a view similar to that of the situation calculus [MH69], but instead of dealing with historical states (situations), the event calculus deals with local events and their consequences.

#### 3.2.2.2 The Event ontology

Our Event ontology considers the existence of event *tokens* [Gal91, VR96] i.e. events are modelled as first-class entities. Regarding the ontological status of our event tokens, we adopt Allen and Fergusson' view [AF94]:

> "Events are primarily linguistic or cognitive in nature. That is, the world does not really contain events. Rather, events are the way by which agents classify certain useful and relevant patterns of change."

We define our event concept as the way by which cognitive agents classify arbitrary regions of space–time. This concept essentially captures an *act of classification*. We will study a broad range of examples in the following sections.

$$\mathsf{event{:}Event}(E) \supset \mathsf{owl{:}Thing}(E) \tag{3.16}$$

Our event concept is related to a space–time extent:

$$\mathsf{event{:}place}(E, P) \supset \mathsf{event{:}Event}(E) \wedge \mathsf{event{:}Place}(P) \tag{3.17}$$

$$\mathsf{event{:}time}(E, T) \supset \mathsf{event{:}Event}(E) \wedge \mathsf{time{:}TemporalEntity}(T) \tag{3.18}$$

The event:time predicate relates an event token to a temporal entity. When using the Timeline ontology described in § 3.2.1 along with the Event ontology, we can for example classify a particular region of an audio signal. In the following, we classify a region of a discrete signal timeline as holding a singing voice:

$$\mathsf{:SingEvent}(E) \supset \mathsf{event{:}Event}(E)$$

$$\mathsf{:SingEvent}(e) \wedge \mathsf{event{:}time}(e, t) \wedge \mathsf{tl{:}Interval}(t) \wedge \mathsf{tl{:}start}(t, 13345) \wedge \mathsf{tl{:}end}(t, 32578) \wedge \mathsf{tl{:}timeline}(t, tl)$$

Events are related to agents i.e. active participants. For example, agents of a performance may include performers, listeners or sound engineers. Events are also related to factors. Factors

[4]Shanahan gives a comprehensive ontology of the event calculus in [Sha99].

of a performance may include the musical instruments or the printout of the score used. Finally, events are related to products. A physical sound would be the product of a performance.

$$\mathsf{event{:}agent}(E,T) \vee \mathsf{event{:}factor}(E,T) \vee \mathsf{event{:}product}(E,T) \supset \mathsf{event{:}Event}(E) \wedge \mathsf{owl{:}Thing}(T) \tag{3.19}$$

$$\mathsf{event{:}Factor}(F) \equiv \exists E.\mathsf{event{:}factor}(E,F) \tag{3.20}$$

$$\mathsf{event{:}Product}(P) \equiv \exists E.\mathsf{event{:}product}(E,P) \tag{3.21}$$

$$\mathsf{event{:}Agent}(A) \equiv \exists E.\mathsf{event{:}agent}(E,A) \tag{3.22}$$

We also introduce a mechanism to decompose complex events into simpler events, each of which can carry part of the information pertaining to the whole:

$$\mathsf{event{:}sub\_event}(E,SE) \supset \mathsf{event{:}Event}(E) \wedge \mathsf{event{:}Event}(SE) \tag{3.23}$$

For example, a musical performance might include several performers playing different instruments at different times. We can decompose the overall performance into several events, corresponding to particular performers and their musical instruments. The following example describes two performers playing two different instruments:

$$\begin{aligned} &\mathsf{event{:}Event}(p) \wedge (\exists p, p_1, p_2. \\ &\quad \mathsf{event{:}sub\_event}(p,p_1) \wedge \mathsf{event{:}sub\_event}(p,p_2) \\ &\quad \wedge \mathsf{event{:}agent}(p_1, \mathsf{chet\_baker}) \wedge \mathsf{event{:}factor}(p_1, \mathsf{trumpet}) \\ &\quad \wedge \mathsf{event{:}agent}(p_2, \mathsf{charlie\_parker}) \wedge \mathsf{event{:}factor}(p_2, \mathsf{saxophone})) \end{aligned} \tag{3.24}$$

Our event model can be depicted as in Figure 3.4. This ontology is primarily available in the OWL-DL sub-language of the Ontology Web Language, and published at [RA06a].

Our event definition is more flexible than the definition used in the ABC framework described in § 3.1.3, as it can also handle acts of classification of temporal objects, e.g. annotating a musical work, a performance or an audio signal. Moreover, other aspects of ABC are difficult to apply in a music context:

- The act of musical composition – An ABC workflow cannot result in the creation of a work. This highlights a conceptual difference between works in FRBR (which are created) and works in ABC. We will discuss musical works further in § 3.3.2.2.
- Specific contributions to an event – ABC requires a new "action" concept for modelling them.
- Arrangements, sounds, signals, etc. – ABC does not include any abstraction layer between work and manifestation.
- Describing a simple performance involving two performers requires an event, two actions, two situations (before and after the performance), several contexts detailing what is holding

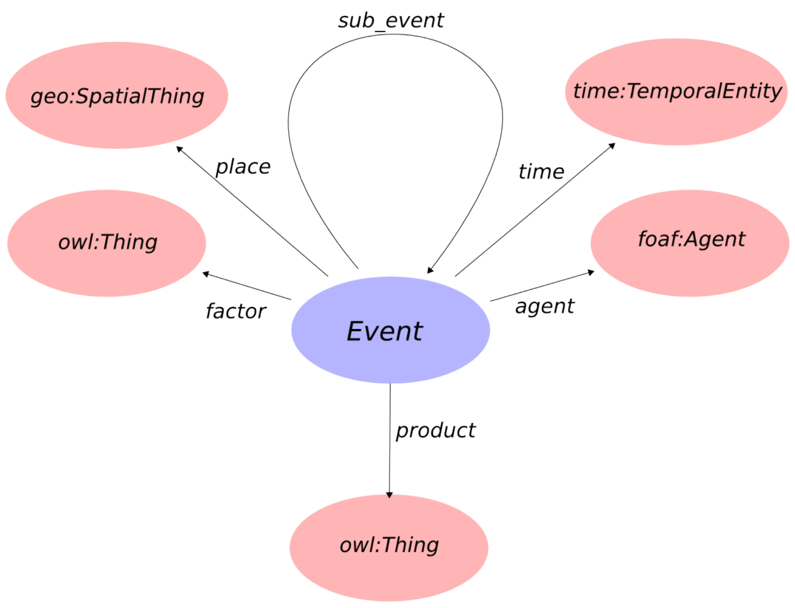


Figure 3.4: Depiction of the event concept and related predicates.

in each of the situations. Moreover, what would be the difference between the situation before the performance and the situation after? Our event model is simpler (one concept and six properties), yet still able to support complex music production workflows, as shown in § 3.3.2.

## 3.3 The Music Ontology

Although the above described ontologies provide a good basis for music-related information, we need a simpler way to address common concepts in this domain. This is the purpose of the Music Ontology [RASG07], providing a number of music-specific refinements of these ontologies.

In this section, we describe this ontology, as well as its relationships to the ontologies described above. We divide Music Ontology terms in three clusters dealing respectively with editorial information, production information and temporal annotations/event decomposition. At the end of the section, we describe ontologies that build on Music Ontology terms.

### 3.3.1 Editorial data

The first cluster of terms in the Music Ontology deals with editorial information. It aims at expressing information such as:

> The first track on the "Nevermind" album by Nirvana is called "Smells Like Teen Spirit". An OGG version of this track is available as `http://example.com/track.ogg`.

Such a statement involves several concepts: "album", "track", "band" and "audio file". We first give a review of the ontologies we use as a basis for such concepts, and we define the terms we need to cover such information.

#### 3.3.1.1 FOAF

In order to deal with persons, relationships between persons, groups of people and organisation, we consider using the Friend-of-a-Friend ontology (FOAF[5] [BM07]) as a basis for such concepts. FOAF defines a main agent concept. We relate this concept to our event ontology in § 3.2.2.2. We consider agents as entities that can be involved in an event as defined in § 3.2.2. Persons, groups and organisations are then concepts subsuming the agent concept:

$$\mathsf{event{:}Agent}(P) \equiv \mathsf{foaf{:}Agent}(P) \tag{3.25}$$

$$\mathsf{foaf{:}Person}(P) \supset \mathsf{foaf{:}Agent}(P) \tag{3.26}$$

$$\mathsf{foaf{:}Group}(P) \supset \mathsf{foaf{:}Agent}(P) \tag{3.27}$$

$$\mathsf{foaf{:}Organization}(P) \supset \mathsf{foaf{:}Agent}(P) \tag{3.28}$$

FOAF defines a number of relationships between these concepts. For example, a person can know another person, and a person can be part of a group:

$$\mathsf{foaf{:}knows}(P, Q) \supset \mathsf{foaf{:}Person}(P) \wedge \mathsf{foaf{:}Person}(Q) \tag{3.29}$$

$$\mathsf{foaf{:}member}(G, P) \supset \mathsf{foaf{:}Group}(G) \wedge \mathsf{foaf{:}Person}(P) \tag{3.30}$$

Other ontologies subsume FOAF terms, such as the Relationship ontology [DJ05] which defines a number of ways a person could be related to another person, by subsuming the foaf:knows relationship.

#### 3.3.1.2 FRBR Items and Manifestations

As described in § 3.1.1.2, the FRBR model provides two abstractions which can act as a basis for editorial information. An item is a concrete entity, such as a copy of a particular album. A manifestation encompasses all the physical objects that bear the same characteristics, both in their actual content and in their physical form. For example, a CD album is a manifestation in FRBR terms. These two concepts are related through a predicate, which ties a manifestation to several items:

$$\mathsf{frbr{:}exemplar}(M, I) \supset \mathsf{frbr{:}Manifestation}(M) \wedge \mathsf{frbr{:}Item}(I) \tag{3.31}$$

#### 3.3.1.3 Music Ontology terms for editorial information

We define a number of music-related terms specifying these different concepts. We define the concepts of music artists (which are considered as agents within the FOAF ontology in order to cover personæ), bands, musical records, tracks and record labels[6]:

[5]We note that FOAF is not a valid OWL-DL ontology. Hence, we redefine the terms we use within our ontological framework in this section.

[6]The list of subsuming concepts for musical items and manifestations is incomplete.

$$\mathsf{mo{:}MusicGroup}(G) \supset \mathsf{foaf{:}Group}(G) \tag{3.32}$$

$$\mathsf{mo{:}MusicArtist}(G) \supset \mathsf{foaf{:}Agent}(G) \tag{3.33}$$

$$\mathsf{mo{:}CorporateBody}(C) \supset \mathsf{foaf{:}Organization}(C) \tag{3.34}$$

$$\mathsf{mo{:}Label}(L) \supset \mathsf{mo{:}CorporateBody}(L) \tag{3.35}$$

$$\mathsf{mo{:}Record}(M) \vee \mathsf{mo{:}Track}(M) \supset \mathsf{frbr{:}Manifestation}(M) \tag{3.36}$$

$$\mathsf{mo{:}track}(R, T) \supset \mathsf{mo{:}Record}(R) \wedge \mathsf{mo{:}Track}(T) \tag{3.37}$$

$$\mathsf{mo{:}AudioFile}(I) \vee \mathsf{mo{:}CD}(I) \vee \mathsf{mo{:}Vinyl}(I) \vee \mathsf{mo{:}Cassette}(I) \vee \mathsf{mo{:}Bittorrent}(I) \supset \mathsf{frbr{:}Item}(I) \tag{3.38}$$

The example mentioned in the introduction of this section can be expressed as:

$$\begin{aligned} &\mathsf{mo{:}MusicGroup}(\mathsf{nirvana}) \wedge \mathsf{foaf{:}made}(\mathsf{nirvana}, \mathsf{nevermind}) \\ &\quad \wedge \mathsf{mo{:}track}(\mathsf{nevermind}, \mathsf{smells\_like\_teen\_spirit}) \\ &\quad \wedge \mathsf{mo{:}track\_number}(\mathsf{smells\_like\_teen\_spirit}, 1) \\ &\quad \wedge \mathsf{frbr{:}exemplar}(\mathsf{smells\_like\_teen\_spirit}, \mathsf{http://example.com/track.ogg}) \end{aligned} \tag{3.39}$$

### 3.3.2 Music production workflow

The second cluster of terms in the Music Ontology deals with music production workflow information. It aims at expressing information such as:

> A performance by Julian Lloyd Webber of the “Air on the G String” arrangement made in the 19th century of the second movement of Johann Sebastian Bach’s Orchestral Suite No 3.

Such information can be used alongside editorial information, which makes it possible to describe a publication of a recording of this performance.

#### 3.3.2.1 Event, FRBR Expressions and Works, and physical time

We consider using the event concept as defined in the Event ontology described in § 3.2.2 for describing music production workflows. Events can be used to relate different agents, FRBR expressions and works together in complex workflows, through the use of the event:agent, event:factor and event:product predicates. For example, an expression corresponding to a sound would be the product of a performance event. A sound can then be the factor of a recording event, which would produce an audio signal.

Moreover, the Timeline ontology and more particularly time intervals defined on the universal timeline allows us to capture the temporal extent of such workflow events.

#### 3.3.2.2 Music Ontology terms for production workflow information

We define several music-related events as follows:

$$\mathsf{mo{:}MusicalEvent}(E) \supset \mathsf{event{:}Event}(E) \tag{3.40}$$

$$\mathsf{mo{:}Performance}(E) \vee \mathsf{mo{:}Recording}(E) \vee \mathsf{mo{:}Composition}(E) \vee \mathsf{mo{:}Arrangement}(E) \vee \mathsf{mo{:}Instrumentation}(E) \vee \mathsf{mo{:}Orchestration}(E) \supset \mathsf{mo{:}MusicalEvent}(E) \tag{3.41}$$

Event-specific *roles*, e.g. conducting a performance or engineering a recording session, are formalised as follows:

$$\mathsf{mo{:}conductor}(E,P) \supset \mathsf{mo{:}Performance}(E) \wedge \mathsf{event{:}Agent}(P) \wedge \mathsf{event{:}agent}(E,P) \tag{3.42}$$

$$\mathsf{mo{:}Conductor}(P) \equiv \exists E.\mathsf{mo{:}conductor}(E,P) \tag{3.43}$$

$$\mathsf{mo{:}performer}(E,P) \supset \mathsf{mo{:}Performance}(E) \wedge \mathsf{event{:}Agent}(P) \wedge \mathsf{event{:}agent}(E,P) \tag{3.44}$$

$$\mathsf{mo{:}Performer}(P) \equiv \exists E.\mathsf{mo{:}performer}(E,P) \tag{3.45}$$

$$\mathsf{mo{:}engineer}(E,P) \supset \mathsf{mo{:}Recording}(E) \wedge \mathsf{event{:}Agent}(P) \wedge \mathsf{event{:}agent}(E,P) \tag{3.46}$$

$$\mathsf{mo{:}Engineer}(P) \equiv \exists E.\mathsf{mo{:}engineer}(E,P) \tag{3.47}$$

$$\mathsf{mo{:}composer}(E,P) \supset \mathsf{mo{:}Composition}(E) \wedge \mathsf{event{:}Agent}(P) \wedge \mathsf{event{:}agent}(E,P) \tag{3.48}$$

$$\mathsf{mo{:}Composer}(P) \equiv \exists E.\mathsf{mo{:}composer}(E,P) \tag{3.49}$$

For example, conductors are individuals that conducted a performance. We define other roles, e.g. violinist, singer or producer, in the same way.

We also specify that such workflow events are linked to a time interval supported by the universal timeline:

$$\mathsf{mo{:}MusicalEvent}(E) \wedge \mathsf{tl{:}time}(E,T) \supset \mathsf{tl{:}UniversalInterval}(T) \tag{3.50}$$

where $\mathsf{tl{:}UniversalInterval}(T) \equiv \mathsf{tl{:}Interval}(T) \wedge \mathsf{tl{:}timeline}(T, \mathsf{tl{:}universaltimeline})$

We define music-related works as follows:

$$\mathsf{mo{:}MusicalWork}(W) \supset \mathsf{frbr{:}Work}(W) \tag{3.51}$$

$$\mathsf{mo{:}performance\_of}(P,W) \equiv \mathsf{mo{:}Performance}(P) \wedge \mathsf{mo{:}MusicalWork}(W) \wedge \mathsf{event{:}factor}(P,W) \tag{3.52}$$

We adopt a view inspired by Levinson [Lev80] for our ontology of musical works. A musical work is defined by the union of the three following criteria:

> "Musical works must be such that they do not exist prior to the composer's compositional activity, but are brought into existence by that activity."

> "Musical works must be such that composers composing in different musico-historical

contexts who determine identical sound structures invariably compose distinct musical works."

"Musical works must be such that specific means of performance or sound production are integral to them."

The definition of a musical work therefore depends on the act of composition—the composition event defined earlier. Such an event includes information about the musico-historical context at its particular time and place. Therefore, two different acts of composition would lead to two distinct musical works. The identity criteria for the musical work does not lie solely in its structure. Moreover, a musical work is also a factor of its performances.

Of course, such a definition does not solve all the issues raised by Bohlman [Boh99]: what about musical works in cultures considering that they exist 'out there' and just wait to be discovered (as in some Eskimo cultures of north-western North America)? Also, Dodd [Dod00] argues that abstract objects cannot be created, as abstract objects cannot be involved in causal relations, and the act of creation is a causal relation. This relates to the problem mentioned in § 3.2.2.2 of considering composition events in the ABC model. Situations do not contain abstract objects, and we therefore cannot distinguish a situation in which a work does not exist from a situation in which it does. These views tend towards *musical Platonism* i.e. all musical works exist independently of the composer, who just selects them [Sha95]. The act of composition then becomes an act of creative discovery. However, the flexible ontological status of our event concept, not necessarily capturing a causal relation, allows us to adapt to such views. The act of creative discovery then becomes our composition event. However, the identity criteria for musical works changes: two composers can potentially "discover" the same musical work. We can also adapt our ontological framework to the view expressed by Caplan and Matheson [CM04]. A musical work then becomes the link between "Beethoven's 1804-1808 compositional activity, performances of the work, and such things as copies of the score and recordings". In this view, the structure of the work is then not anymore an attribute of the work. Its different expressions entail it.

We define a number of music-related expressions:

$$\mathsf{mo{:}Sound}(E) \vee \mathsf{mo{:}Signal}(E) \vee \mathsf{mo{:}Score}(E) \supset \mathsf{frbr{:}Expression}(E) \tag{3.53}$$

Sounds correspond to spatio-temporal acoustic fields. Signals correspond to a function from time to numerical values. We will detail the signal concept in § 3.3.3.1. Scores correspond to symbolic representations. We will study an extension of our ontological framework making use of the score concept in § 3.3.4.3.

A depiction of the main concepts and relationships in a typical Music Ontology workflow is in Figure 3.5

The example mentioned in the introduction of § 3.3.2 can be expressed as:

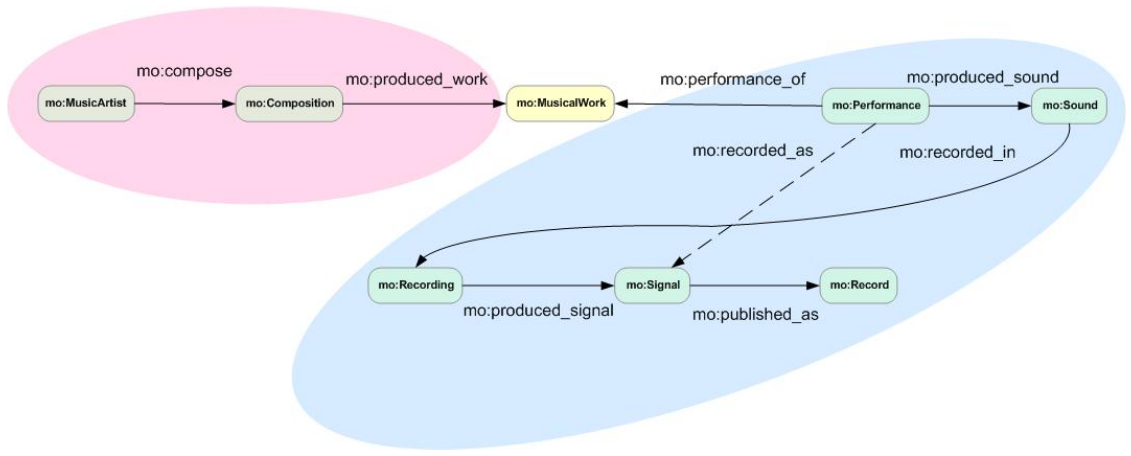


Figure 3.5: Depiction of the main concepts and relationships involved in describing a simple Music Ontology workflow.

$$\begin{aligned}&\mathsf{mo{:}Performance}(p) \wedge \mathsf{mo{:}performer}(p, \mathsf{webber}) \wedge \mathsf{mo{:}performance\_of}(p, \mathsf{air\_on\_the\_G\_string}) \\ &\quad \wedge \mathsf{mo{:}Instrumentation}(i) \wedge \mathsf{event{:}product}(i, \mathsf{air\_on\_the\_G\_string}) \\ &\quad \wedge \mathsf{event{:}time}(i, t_1) \wedge \mathsf{tl{:}during}(t_1, \mathsf{nineteenth\_century}) \\ &\quad \wedge \mathsf{event{:}factor}(i, m) \wedge \mathsf{mo{:}Movement}(m) \wedge \mathsf{mo{:}movement}(\mathsf{bwv1068}, m) \wedge \mathsf{mo{:}number}(m, 2) \\ &\quad \wedge \mathsf{mo{:}Composition}(c) \wedge \mathsf{mo{:}produced\_work}(c, \mathsf{bwv1068}) \wedge \mathsf{mo{:}composer}(c, \mathsf{jsbach}) \\ &\quad \wedge \mathsf{tl{:}time}(c, t_2) \wedge \mathsf{tl{:}during}(t_2, 1717)\end{aligned} \tag{3.54}$$

### 3.3.3 Signals, temporal annotations and event decomposition

The third cluster of Music Ontology terms deals with signals, temporal annotations and event decomposition (following the mechanism described in § 3.2.2). This part of the Music Ontology is therefore entirely built upon the Timeline and the Event ontologies, respectively described in § 3.2.1 and § 3.2.2.

#### 3.3.3.1 Signals and timelines

We define signal-related terms, specifying the mo:Signal concept defined in § 3.3.2.2. These terms encompass audio signals and derived signals e.g. a power spectrogram, as defined in § $A$.2.

$$\mathsf{af{:}AudioSignal}(s) \supset \mathsf{mo{:}Signal}(s) \tag{3.55}$$

$$\mathsf{af{:}Spectrogram}(s) \supset \mathsf{mo{:}Signal}(s) \tag{3.56}$$

(3.57)

Signals are related to a corresponding temporal extent, which is itself defined on a timeline:

$$\mathsf{mo{:}signal\_time}(S, T) \supset \mathsf{mo{:}Signal}(S) \wedge \mathsf{tl{:}Interval}(T) \tag{3.58}$$

Now, we can relate signals in complex ways, through the timeline map concept defined in

§ 3.2.1.2. For example, the temporal relationship between the spectrogram of an audio signal and the audio signal itself would be defined as in:

$$\begin{aligned}&\mathsf{af{:}Spectrogram}(p) \wedge \mathsf{mo{:}signal\_time}(p, t_p) \wedge \mathsf{tl{:}timeline}(t_p, l_p) \\ &\quad \wedge \mathsf{af{:}AudioSignal}(s) \wedge \mathsf{mo{:}signal\_time}(s, t_s) \wedge \mathsf{tl{:}timeline}(t_s, l_s) \\ &\quad \wedge \mathsf{tl{:}UniformWindowingMap}(m) \wedge \mathsf{tl{:}domainTimeLine}(m, l_s) \wedge \mathsf{tl{:}rangeTimeLine}(m, l_p) \\ &\quad \wedge \mathsf{tl{:}hopSize}(m, 128) \wedge \mathsf{tl{:}windowLength}(m, 256)\end{aligned} \tag{3.59}$$

where the predicates $\mathsf{tl{:}hopSize}(m, 128)$ and $\mathsf{tl{:}windowLength}$ specify the hop size and the window size used in the windowing process. By keeping track of such timeline maps, we are able to draw links from one signal to the other. An interval defined on the spectrogram timeline is related to the corresponding interval on the signal timeline.

#### 3.3.3.2 Temporal annotations

As explained in § 3.2.2.2, we can use the Event Ontology to classify regions of timelines associated with signals, performances, scores, etc. The cognitive agents mentioned in § 3.2.2.2 might be artificial. Classifications can be derived by algorithms. Several extensions of the Music Ontology define classifiers subsuming the event concept, as illustrated § 3.3.4.2, § 3.3.4.3 and § 3.3.4.5.

An example of a simple annotation is the following, a structural segmentation of the signal $s$ corresponding to the "Can't Buy Me Love" song from the Beatles, as issued on the "A Hard Day's Night" 1964 Parlophone record. We first define two classifiers.

$$\mathsf{pop{:}Chorus}(e) \supset \mathsf{event{:}Event}(e) \tag{3.60}$$

$$\mathsf{pop{:}Verse}(e) \supset \mathsf{event{:}Event}(e) \tag{3.61}$$

We then classify regions of $s$.

$$\begin{aligned}&\mathsf{mo{:}Signal}(s) \wedge \mathsf{mo{:}signal\_time}(s, t) \wedge \mathsf{tl{:}duration}(t, 134) \wedge \mathsf{tl{:}timeline}(t, l)^{7} \\ &\quad \wedge \mathsf{pop{:}Chorus}(c_1) \wedge \mathsf{event{:}time}(c_1, t_1) \wedge \mathsf{tl{:}timeline}(t_1, l) \wedge \mathsf{tl{:}start}(t_1, 0) \wedge \mathsf{tl{:}duration}(t_1, 9) \\ &\quad \wedge \mathsf{pop{:}Verse}(v_1) \wedge \mathsf{event{:}time}(v_1, t_2) \wedge \mathsf{tl{:}timeline}(t_2, l) \wedge \mathsf{tl{:}start}(t_2, 9) \wedge \mathsf{tl{:}duration}(t_2, 33) \\ &\quad \wedge \mathsf{pop{:}Chorus}(c_2) \wedge \mathsf{event{:}time}(c_2, t_3) \wedge \mathsf{tl{:}timeline}(t_3, l) \wedge \mathsf{tl{:}start}(t_3, 41) \wedge \mathsf{tl{:}duration}(t_3, 13)\end{aligned} \tag{3.62}$$

#### 3.3.3.3 Event decomposition

As described in § 3.2.2, complex events can be decomposed in several simpler events, each of them holding a bit of information pertaining to the whole. We already saw at the end of § 3.2.2 an example of splitting a musical performance into two sub-performances corresponding to two performers playing two different instruments. We can also apply the same mechanism to other events defined within the Music Ontology, such as recording events. For example, we use event

[7] The coordinate system used on the timeline $tl$ corresponds to the number of seconds.

decomposition in the following example to express the position and the type of two microphones used in a studio recording:

$$\begin{aligned}&\mathsf{mo{:}Recording}(r) \wedge \mathsf{event{:}sub\_event}(r, r_1) \wedge \mathsf{event{:}sub\_event}(r, r_2) \\ &\quad\wedge \mathsf{event{:}place}(r_1, p_1) \wedge \mathsf{event{:}place}(r_2, p_2) \\ &\quad\wedge \mathsf{event{:}factor}(r_1, \mathsf{sm\_57}) \wedge \mathsf{event{:}factor}(r_2, \mathsf{pg\_52})\end{aligned} \tag{3.63}$$

We can go further by describing the locations $p1$ and $p2$. They can either be defined relatively to other objects in the studio (e.g. “at the top left of the guitar amplifier” or “kick drum microphone”) or through exact absolute locations (latitude, longitude and altitude).

A composition event can also be decomposed in several sub-events, expressing different influences in different places over time, which eventually led to the creation of a musical work.

#### 3.3.3.4 Web ontologies

The Timeline, Event and Music ontologies define our Music Ontology framework. They are available as sets of web identifiers and corresponding structured representation. These web identifiers are clustered in three namespaces, one per ontology:

- `http://purl.org/NET/c4dm/timeline.owl#` for the Timeline ontology [RA06b] ;
- `http://purl.org/NET/c4dm/event.owl#` for the Event ontology [RA06a] ;
- `http://purl.org/ontology/mo/` for the Music ontology [RASG07].

The structured representations of terms within these ontologies provide the links between the ontologies. We get, among others, the following RDF statements when accessing `http://purl.org/ontology/mo/Performance`:

```
mo:Performance
  a owl:Class;
  rdfs:subClassOf event:Event .
```

Then, we can follow for example the link to the `event:Event` concept, which leads us to the definition in § 3.2.2.2.

### 3.3.4 Available Extensions

The Music Ontology provides a core knowledge representation framework for music-related information, and provides many extension points. For example, the ontology itself provides only very basic instrument and genre terms, but can be extended by using an instrument taxonomy and a genre taxonomy. In this section we study such extensions of the Music Ontology i.e. web ontologies that link to terms within the Music Ontology framework.

#### 3.3.4.1 Instrument taxonomy

Herman [Her07] published an instrument taxonomy. The taxonomy is organised using the Simple Knowledge Organisation System (SKOS [MMWB05]). The top-level concept in this taxonomy is the instrument concept defined in the Music Ontology. The following is a small excerpt from this taxonomy of musical instruments:

```
mit:Fipple_flutes
  a skos:Concept;
  skos:narrower mit:Bansuri;
  skos:narrower mit:Shakuhachi;
  skos:narrower mit:Slide_whistle;
  skos:narrower mit:Tin_whistle;
  skos:narrower mit:Traverse_flute;
  skos:narrower mit:Vertical_flute;
  skos:narrower mit:Willow_flute;
  skos:narrower mit:Recorder;
  skos:prefLabel "Fipple flutes".

mit:Traverse_flute
  a skos:Concept;
  skos:narrower mit:Alto_flute;
  skos:narrower mit:Piccolo;
  skos:narrower mit:Sao_truc;
  skos:prefLabel "Traverse flute".
```

#### 3.3.4.2 Chord Ontology

We published a web ontology for expressing musical chords [SRM07] . This ontology is grounded on the symbolic representation of musical chords described by Harte et al. [HSAG05]. A chord is defined, in the most general case, by a root note and some constituent intervals. A chord inversion may be indicated by specifying which interval is the bass. The ontology also defines a way to build chords on top of other chords. The concepts and relationships in this chord ontology are depicted in Figure 3.6.

For example, we can use this ontology to represent a D sharp minor with added ninth and missing flat third, over the fifth (this would be the chord depicted in Figure 3.7). Such a chord would be represented as in Figure 3.8.

We designed a namespace able to provide RDF representations using this ontology for chords expressed within Harte's notation. The example depicted in Figure 3.8 can be accessed at the URI `http://purl.org/ontology/chord/symbol/Ds:min7(*b3,9)/5`, which itself corresponds to the chord `Ds:min7(*b3,9)/5` in Harte's notation.

The ontology also defines a chord event concept, subsuming the event concept in the Event ontology described in § 3.2.2. It also defines a sub-property of the event:factor property, in order to attach a particular chord to such an event. We can then annotate timelines as described in § 3.3.3.2 with corresponding chords. We now take the same example as in § 3.3.3.2 ("Can't Buy Me Love" by the Beatles) and describe the first two chords (we make use of the above mentioned

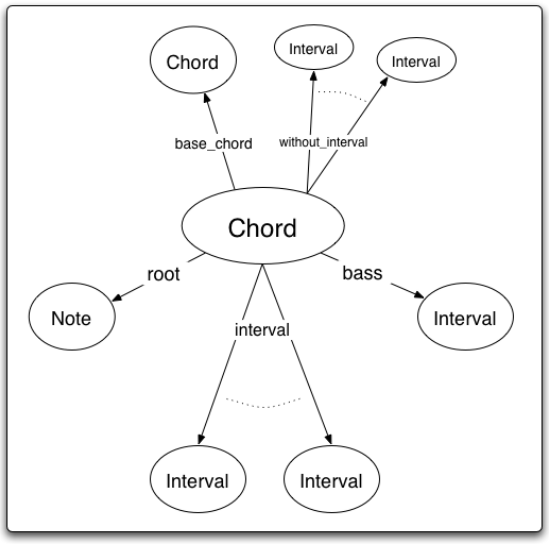


Figure 3.6: Depiction of the main concepts and relationships in the Chord Ontology

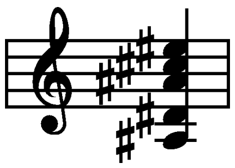


Figure 3.7: D sharp minor with added ninth and missing flat third, over the fifth

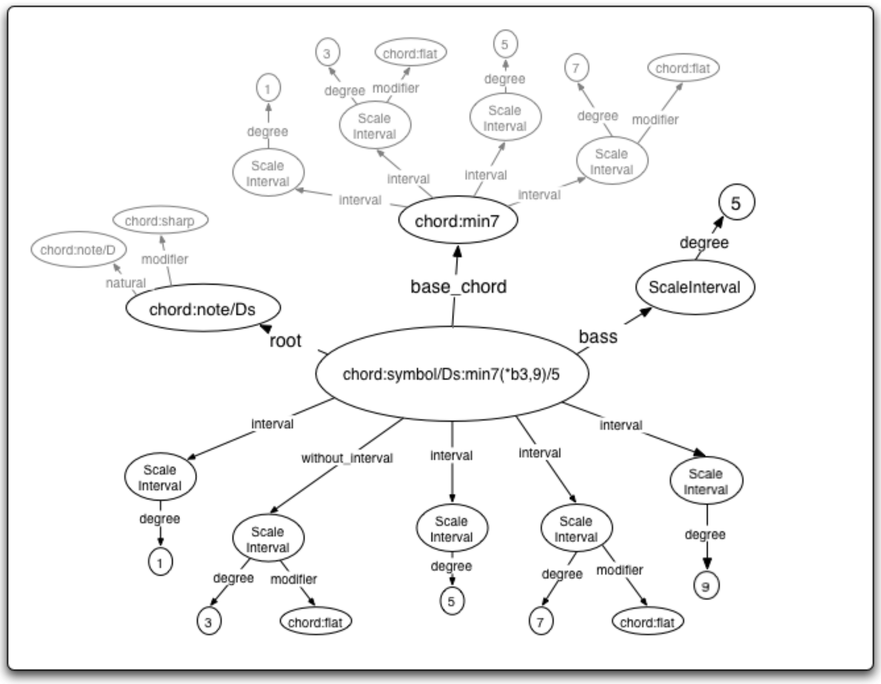


Figure 3.8: Representation of a D sharp minor with added ninth and missing flat third, over the fifth. Shaded parts of the graph correspond to RDF statements that may be accessed by accessing the corresponding web identifiers.

chord namespace here):

```
<http://dbtune.org/musicbrainz/resource/signal/eb20ee61-414f-4eee-8dce-
190db516a466>
   a mo:Signal;
   rdfs:label "Signal for Can’t Buy Me Love in the Musicbrainz dataset";
   mo:time [
      tl:onTimeLine :tl
   ].

:tl a tl:RelativeTimeLine.

:ce1
   a chord:ChordEvent;
   rdfs:label "C chord";
   chord:chord <http://purl.org/ontology/chord/symbol/C>;
   event:time :t1.

:t1
   a tl:Interval;
   tl:onTimeLine :tl;
   tl:start "P0.4595427S"^^xsd:duration;
   tl:duration "P1.550498S"^^xsd:duration.

:ce2
   a chord:ChordEvent;
   rdfs:label "E minor chord";
   chord:chord <http://purl.org/ontology/chord/symbol/E:min>;
   event:time :t2.

:t2
   a tl:Interval;
   tl:onTimeLine :tl;
   tl:start "P1.550498S"^^xsd:duration;
   tl:duration "P2.850816S"^^xsd:duration.
```

#### 3.3.4.3 Symbolic Ontology

We also devised an extension of the Music Ontology dealing with musical events in a symbolic context [Rai07f]. The ontology uses as a basis the work of Smaill et al. [SWH93] mentioned in § 3.1.2.1. It therefore focuses on performance information, although it also allows temporal information to be under-specified, as in Balaban’s framework [Bal96].

The ontology defines a number of concepts subsuming the event concept in the Event ontology described in § 3.2.2:

$$\mathsf{so{:}SymbolicEvent}(E) \supset \mathsf{event{:}Event}(E) \tag{3.64}$$
$$\mathsf{so{:}NoteEvent}(E) \supset \mathsf{so{:}SymbolicEvent}(E) \tag{3.65}$$
$$\mathsf{so{:}RestEvent}(E) \supset \mathsf{so{:}SymbolicEvent}(E) \tag{3.66}$$

The ontology also defines a particular type of timeline, supporting such symbolic representations and addressed canonically in number of beats:

$$\mathsf{so{:}ScoreTimeLine}(L) \supset \mathsf{tl{:}TimeLine}(L) \tag{3.67}$$
$$\mathsf{tl{:}Interval}(T) \land \mathsf{tl{:}timeline}(T, L) \land \mathsf{so{:}ScoreTimeLine}(L) \equiv \mathsf{so{:}ScoreInterval}(T) \tag{3.68}$$
$$\mathsf{so{:}SymbolicEvent}(E) \land \mathsf{event{:}time}(E, T) \equiv \mathsf{so{:}ScoreInterval}(T) \tag{3.69}$$

The ontology subsumes the note and rest events by a number of concepts implicitly specifying their relative durations, for example:

$$\mathsf{so{:}QuarterNote}(N) \equiv \mathsf{event{:}time}(N, T) \land \mathsf{so{:}Quarter}(T) \tag{3.70}$$
$$\mathsf{so{:}Quarter}(T) \equiv \mathsf{so{:}ScoreInterval}(T) \land \mathsf{tl{:}duration}(T, 0.25) \tag{3.71}$$

Using the Allen's relationships described in § 3.2.1, we can use this ontology to express the relative ordering of note events. This ontology is then expressive enough to express the global position of a musical event in a symbolic representation.

The ontology also defines a number of ways to gather such events into complex wholes (using the event decomposition mechanism explained in § 3.2.2):

$$\mathsf{so{:}Bar}(E) \lor \mathsf{so{:}Tie}(E) \lor \mathsf{so{:}Slur}(E) \lor \mathsf{so{:}Voice}(E) \lor \mathsf{so{:}Motif}(E) \supset \mathsf{event{:}Event}(E) \tag{3.72}$$

The bar, tie and slur events capture the corresponding concepts in symbolic music representation. The voice and motif concepts correspond to more arbitrary clusters of symbolic events. We can also use the representation of temporal aggregates defined by Pan [Pan07] to specify the time signature of a particular piece (i.e. 'in this piece, a bar event is happening every four beats, and lasts four beats').

Using this ontology, we are able to represent the first bar of Debussy's Syrinx as in Appendix D.

#### 3.3.4.4 Synchronisation and timing

Using the above defined ontology along with our Timeline ontology described in § 3.2.1.2, we can associate "score time" with "real time".

We first define a specific subset of timeline maps, encompassing such synchronisations.

$$\mathsf{so{:}SymbolicTimeLineMap}(M) \supset \mathsf{tl{:}TimeLineMap}(M) \tag{3.73}$$

$$\mathsf{tl{:}domainTimeLine}(M, L) \wedge \mathsf{so{:}SymbolicTimeLineMap}(M) \supset \mathsf{so{:}ScoreTimeLine}(L) \tag{3.74}$$

$$\mathsf{tl{:}rangeTimeLine}(M, L) \wedge \mathsf{so{:}SymbolicTimeLineMap}(M) \supset (\mathsf{tl{:}RelativeTimeLine}(L) \vee \mathsf{tl{:}PhysicalTimeLine}(L)) \tag{3.75}$$

Such timeline maps can be used to relate metrical time on score timelines to real time on corresponding physical or relative timelines. There are lots of different ways to specify such a mapping (a comprehensive review is given in [Hon01]).

Our timeline map concept encompasses "tempo curves", specifying the tempo as a function of the position in the score:

$$\mathsf{so{:}TempoCurve}(M) \supset \mathsf{so{:}SymbolicTimeLineMap}(M) \tag{3.76}$$

Individuals of this concept hold information about the tempo curve itself. However, it has been argued that is is difficult to express such timing information using just tempo curves [DH93, Hon01]. For example, it is impossible to express the differences of timing between multiple parallel voices, or the asynchrony in performing a chord.

Jaffe [Jaf85] describes an approach based around the notion of a "time map", a function relating a score time to a physical time. Different time maps can be constructed for each voice. Intersection points of these time maps constrain where vertical alignment occur.

$$\mathsf{so{:}JaffeTimeMap}(M) \supset \mathsf{so{:}SymbolicTimeLineMap}(M) \tag{3.77}$$

Another interesting feature of our Music Ontology framework is that we can also define such mappings implicitly, by using the event:factor predicate. In that case, the timeline map itself does not hold any explicit way of mapping a certain time extent on the score timeline to a certain time extent on the physical timeline. Events on the physical timeline (such as "this performer played that particular note") are linked to the corresponding events on the score timeline ("this note is a quaver, and is the first note in the second bar") through event:factor.

#### 3.3.4.5 Ontologies of audio features

Another extension deals with the representation of audio features (higher-level representations derived from the audio signal, as reviewed in Appendix A). Due to the diversity of possibly extractable audio features, extensions dealing with a specific set of audio features are built in a ad-hoc way, on top of the conceptual framework defined by our Audio Features ontology [Rai07a][8]

Our Audio Features ontology defines a timeline-based framework for representing dense or sparse features. For sparse features, we use the Event ontology described in § 3.2.2.2 to classify particular regions of a signal timeline. We define two different classifiers: one for point-based features (e.g. a musical onset) and one for interval-based features (e.g. a chord event, as mentioned in § 3.3.4.2). These classifiers are defined as follows.

[8]The Audio Features ontology also includes example features used by the EASAIER project [eas] and by Vamp plugins [Can].

$$\text{af:Segment}(F) \supset \text{event:Event}(F) \tag{3.78}$$
$$\text{af:Segment}(F) \equiv \exists I.\text{tl:Interval}(I) \wedge \text{event:time}(F, I) \tag{3.79}$$
$$\text{af:Point}(F) \supset \text{event:Event}(F) \tag{3.80}$$
$$\text{af:Point}(F) \equiv \exists I.\text{tl:Instant}(I) \wedge \text{event:time}(F, I) \tag{3.81}$$

For example, classifying an interval as holding a particular chord would be done using the af:ChordSegment concept (which is the same as the chord event concept mentioned in § 3.3.4.2) defined as follows.

$$\text{af:ChordSegment}(S) \supset \text{af:Segment}(S) \tag{3.82}$$
$$\text{co:chord}(S, C) \supset \text{af:ChordSegment}(S) \wedge \text{event:factor}(S, C) \wedge \text{co:Chord}(C) \tag{3.83}$$

The Audio Features ontology also defines a framework for dense features. Some features, such as the spectrogram mentioned in § 3.3.3.1 or the chromagram reviewed in § $A$.3, need to be represented in a more compact way. Dense features are represented as signals. We use the same mechanism as described in § 3.3.3.1 to relate the timeline of the original audio signal and the timeline of the dense feature.

The different concepts and relationships defined in the Audio Features ontology can be depicted as in Figure 3.9.

## 3.4 Discussion

In this section, we discuss the Music Ontology framework we described in this chapter. We discuss its relationships to other web ontologies for the music domain in § 3.4.1, and discuss some points that distinguish it from other music representation systems in § 3.4.2.

### 3.4.1 Comparison with related web ontologies

In the following, we compare our Music Ontology framework with related web ontologies. We show that our ontology is more expressive than any music ontology published on the Web so far. We will perform a quantitative evaluation of our Music Ontology framework in chapter 3.

#### 3.4.1.1 Musicbrainz RDF Schema

The first version of the Musicbrainz[9] web service was RDF-based. The Musicbrainz community therefore designed a small vocabulary [Dod04] specifying the different concepts and relationships used in the outputted RDF. This vocabulary mainly focuses on tracks, albums and artists, which are all handled in the Music Ontology framework as described in § 3.3.1. The Musicbrainz RDF vocabulary does not tackle music production workflows or temporal annotations and is therefore far less expressive than our ontology. For example, there is no way to consider that two tracks correspond to two performances of a same musical work.

[9] `http://musicbrainz.org/`

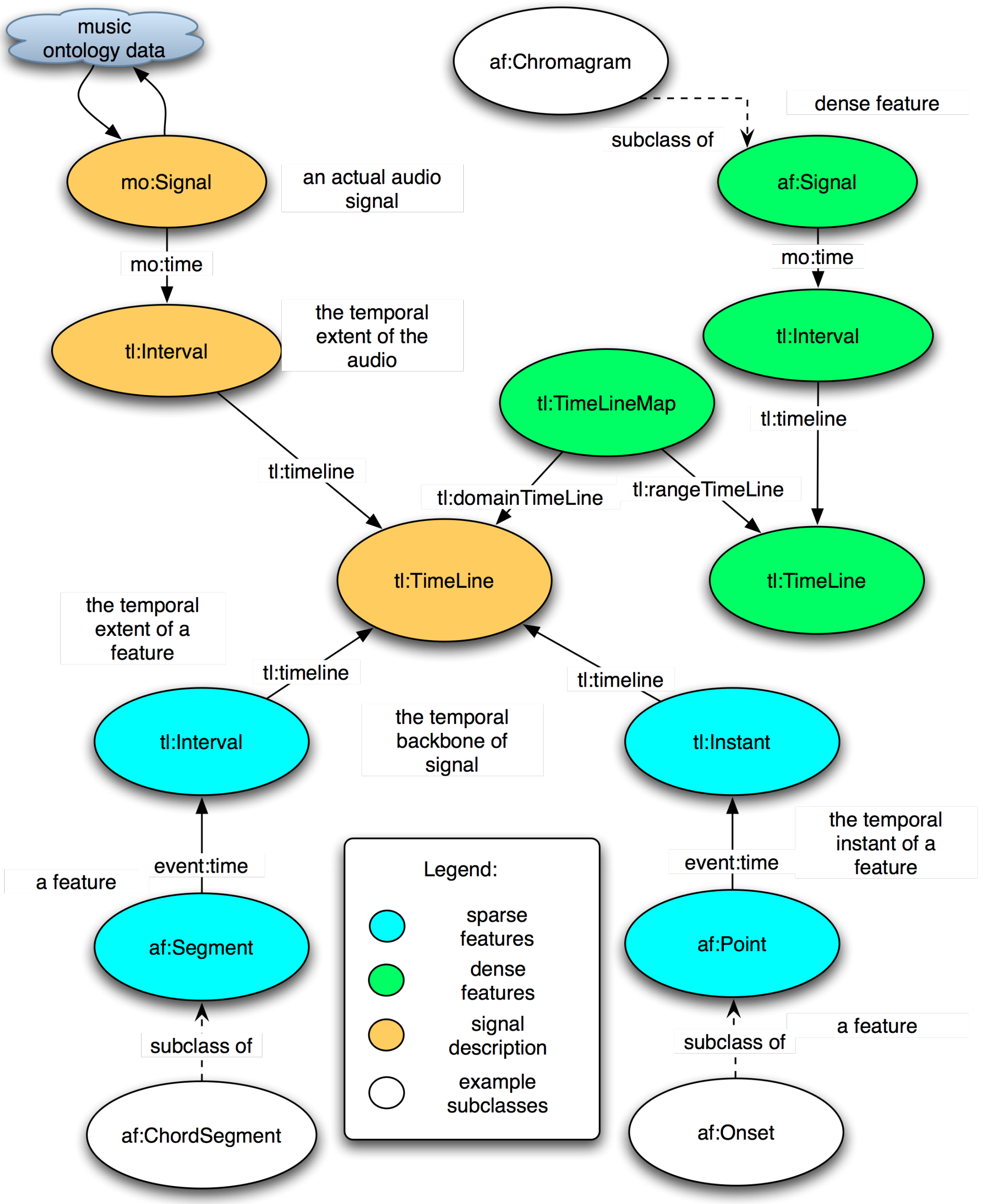


Figure 3.9: Depiction of the concepts and relationships in the Audio Features ontology

#### 3.4.1.2 Nepomuk ID3 Ontology

The Nepomuk ID3 Ontology [MSSvE07] is mainly focused around the concept of an audio item (as is ID3, which it captures as a web ontology). It does not consider any other musical entities. So, for example, it is still impossible to consider that two tracks correspond to two performances of a same musical work.

#### 3.4.1.3 Music Vocabulary

Kanzaki's Music Vocabulary [Kan07] provides terms for describing classical performances. It encompasses musical works, events, instruments and performers. Although our ontology also includes these concepts, the Music Vocabulary modelling is different from ours. In the Music Vocabulary, a musical work is associated to a performance (a "representation" of that work), and the performance is being "presented" in a musical event. The Music Vocabulary does not encompass what happens after the performance, e.g. releasing a recording of the performance. Moreover, it does not handle temporal annotations and event decomposition. For example, it is impossible to state that someone played the violin in a particular performance and the cello in another performance. The person can only be specified as being a violinist and a cellist, but these roles cannot be made specific to a particular performance.

#### 3.4.1.4 SIMAC ontology

The SIMAC music ontology [GC05] is focused around artists, tracks and musical features of tracks, by including a translation of MPEG-7 to a web ontology. It has a similar scope as the ontology developed by Baumann et al. [BKN02][10]. These two ontologies do not deal with musical works, performances, or temporal decompositions at different levels (e.g. score, performance and audio signal). To handle some of these use-cases, the SIMAC music ontology links to the Music Vocabulary described in § 3.4.1.3, and therefore has the same limitations. Moreover, this linkage conflates two levels of abstractions. It considers an audio segment and a section of a musical work as being on the same ontological stand-point. This is an arguable decision, as it makes it impossible to distinguish multiple performances and recordings of a single musical work, or to describe a performance in a different key or meter as the original work.

### 3.4.2 An open music representation system

As Dannenberg points out [Dan93]:

> There can be no "true" representation [of music], just as there can be no closed definition of music.

This highlights a major shortcoming in most current music representation systems. Most of them do not provide the possibility to write arbitrary extensions or to be related to other representation frameworks in other domains, e.g. geographical information or relationships between persons. The Music Ontology provides such an open music representation systems.

[10]These two ontologies are not anymore available on the Web.

#### 3.4.2.1 Web technologies and open music representation systems

The web technologies described in § 2.2 provide us with a good ground for an open music representation system. Each resource in a Music Ontology description (a particular note in a particular arrangement, for example) can be considered as primary and described as such, as opposed to traditional hierarchical 'meta-data' framework, as reviewed in § 3.1.2. Other web ontologies covering other domains, e.g. relationships between people or geographic information, can be used to describe particular aspects of such resources.

Web technologies also provide us with a good ground for ontology extensibility. Any term in a Semantic Web ontology is identified by a URI. Therefore, extending a particular term, refining it, or creating a new predicate linking two concepts in two different ontologies is as easy as just mentioning the corresponding web identifiers in the representation of the new term. The development and the refinement of Music Ontology terms is therefore done in a completely decentralised way.

For example, the instrument taxonomy mentioned in § 3.3.4.1 extends the Music Ontology by mentioning the web identifier mo:Instrument (a musical instrument as defined by the Music Ontology) as its top concept. All sorts of extensions can be defined in the same way, by publishing a web ontology linking to Music Ontology terms. For example, we could publish an audio recording device taxonomy, and stating that these new terms can be used as factors of recording events in the Music Ontology.

Moreover, our ontology might not suit anyone wanting to publish music-related data on the Web. As each term in our ontology is associated to a web identifier, we allow these data publishers to pick up just the terms that they want to use, without having to commit to the entire framework.

This openness can lead to an overall undecidable framework if such an extension takes our ontology outside the $\mathcal{SHOIN}(D)$ boundaries, or it can lead to an inconsistent ontology. We believe this problem is unavoidable in a web context, and reasoning systems operating over structured web data must be able to leave out axioms breaking their decision procedure. Heflin [Hef01] similarly argues that:

> "To scale to the size of the ever growing web, we must either restrict the expressivity of our representation language or use incomplete reasoning algorithms."

#### 3.4.2.2 Extensibility of music production workflows

Our modelling of a music production workflow is easily extensible. We did not consider providing just one way of expressing a music production workflow, as this changes from one use-case to another. For example, a classical music production process and an electro-acoustic music production process cannot be modelled in the same way.

Music production workflows are defined as a set of interconnected events, and the Music Ontology defines some typical events. Adapting to a new production process can be done by defining new specialisations of the event concept and introducing them in the workflow. Such a model is applicable to a wider range of use-cases than the model defined by Garcia [Gar05], which defines three interconnected concepts for handling such workflows: work, performance and manifestation, or other models that commit to a specific production workflow. We therefore recognise the flexibility of the models mentioned in § 3.1.3 to tackle such issues.

Moreover, any kind of unforeseen data can be attached to such events as new factors or agents. For example, we might define a predicate linking a performance to the person in charge of organising it, through a new predicate specifying the `event:agent` predicate.

## 3.5 Summary

In this chapter, we described the Music Ontology framework, providing a set of web identifiers and corresponding representations for a number of concepts and relationships in the music domain. The Music Ontology framework provides a way to publish a wide range of structured music data, including editorial information, information related to the production workflow, temporal annotations or hierarchical symbolic representations.

# Chapter 4

# Evaluation of the Music Ontology framework

In chapter 3, we described our Music Ontology, creating a Semantic Web framework for describing a wide range of musical information, from editorial data to temporal anotations of audio signals. In this chapter, we evaluate our Music Ontology framework.

We validate our approach according to real-world user needs, and also provide some statistical insights for comparison with related representation frameworks, completing the discussion made in § 3.4.1. In order to investigate this comparison further, we also apply in chapter 6 the methodology used by Garcia in [Gar05] — we translate related representation frameworks to Semantic Web ontologies and check if these ontologies can be mapped to ours. If so, our ontology is at least as expressive as related representation frameworks.

We first review previous work on the evaluation of ontologies in § 4.1. We devise our evaluation methodology in § 4.2, measuring how well real-world user needs fit within our Music Ontology framework. We perform in § 4.3 the actual evaluation, and compare several alternatives for each step of our evaluation process. We summarise and discuss the results in § 4.4.

## 4.1 Techniques for ontology evaluation

Brewster et al. argue that standard information retrieval or information extraction evaluation methodologies, using the notion of *precision* and *recall*, are not appropriate for ontology evaluation [BADW04]. We need different evaluation methodologies to evaluate knowledge representation frameworks. Ultimately, we need an ontology evaluation metric, in order to easily assess ontologies and to track their evolution [VS07]. In this chapter, we design such an evaluation metric for music ontologies.

We now review previous work in ontology evaluation. We review different ontology evaluation methodologies in § 4.1.1, § 4.1.2 and § 4.1.3, and explain why they are not suitable for evaluating our Music Ontology framework. We focus on two evaluation paradigms in § 4.1.4 and § 4.1.5 that constitute the basis of our evaluation methodology.

### 4.1.1 Qualitative evaluation

One way to qualitatively evaluate an ontology is to take a set of users and ask them to rate the ontology according to a number of criteria. The OntoMetric evaluation methodology [LTGP04], includes a number of such qualitative metrics. Zhang and Li [ZL08] evaluate two metadata schemes for moving images by asking diverse groups of users to rate usefulness of individual metadata fields according to each generic user task defined by the Functional Requirements for Bibliographic Records (FRBR [otFRfBR98]): finding, identifying, selecting and obtaining.

However, qualitative ontology evaluations have two main problems. It is difficult to choose the right set of users (they could be ontologists, end-users or domain experts), and it is difficult to find an actual scale on which to rate particular criteria of the ontology (what do we mean by a model being "good"?). For these reasons, we do not consider performing a qualitative evaluation of our Music Ontology framework.

### 4.1.2 Structural and ontological metrics

A number of ontology evaluation metrics can be derived automatically. Amongst these, we distinguish between *structural* and *ontological* metrics [VS07].

#### 4.1.2.1 Structural metrics

OWL ontologies, as described in § 2.2.6.3, are defined through an RDF graph detailing the interconnections between concepts and relationships in the domain of interest. This graph can be analysed to derive evaluation metrics. These metrics, evaluating the structure of the graph defining the ontology but not the ontology itself, are called structural metrics. For example, the AKTiveRank system [AB06] includes a metric quantifying the average amount of edges in which a particular node corresponding to a concept is involved. This metric therefore gives an idea of how much detail a concept definition in the evaluated ontology holds.

Such metrics do not capture the intended use of our ontology framework, which is to use the Web as backbone for distributing a wide range of music-related information. We therefore do not consider using structural metrics in our evaluation.

#### 4.1.2.2 Ontological metrics

Ontological metrics evaluate the actual models instead of just their underlying graph structure. The OntoClean methodology [GW02] evaluates modelling choices in ontologies from a philosophical stand-point. It defines a number of criteria that need to be satisfied. For example, a subsumption relationship cannot be drawn between concepts that have different identity criteria — a time interval cannot be a sub-concept of a time duration, as two durations with the same length are the same duration, but two time intervals with the same length are not necessarily the same time interval. OntoClean relates more to ontology engineering than ontology evaluation [VS07]. It can be seen as a set of ontology design guidelines. These guidelines were used when designing the Music Ontology.

### 4.1.3 Similarity measures

If we have access to a "gold-standard" (a canonical model of a particular domain) we can evaluate other ontologies of that domain by measuring their similarities to that canonical model. A set of measures for describing the similarity of different ontologies (both at the lexical and at the conceptual level) is proposed in [MS02]. We do not have such a gold-standard ontology, so this approach can be dismissed for evaluating our Music Ontology framework.

### 4.1.4 Task-based evaluation

Another way of evaluating an ontology is to measure its performance on a specific task [PM04]. A given task is chosen, as well as a corresponding gold-standard for perfect performance. Then, we consider the following errors when trying to fulfill that task in a particular ontology-driven application:

- Insertion errors (some terms in the ontology are not necessary);
- Deletion errors (missing terms);
- Substitution errors (ambiguous or ill-defined terms).

Such a methodology can also be used to compare two ontologies. The evaluation paradigm then becomes similar to the paradigm used in evaluation campaigns such as TREC or MIREX[1], i.e. different frameworks are compared against each other on the basis of their ability to fulfill a particular task.

### 4.1.5 Data-driven ontology evaluation

Brewster et al. provide a data-driven approach for ontology evaluation [BADW04]. They use a corpus of text within the domain modelled by the ontology. They extract terms from it and try to associate them with terms in the ontology to evaluate, which leads to a measure for the domain coverage of the ontology. In order to evaluate the structure of the ontology, they cluster the extracted terms and quantify the extent to which terms in the same cluster are closer in the ontology than terms in different clusters.

## 4.2 A query-driven ontology evaluation methodology

We now devise our methodology for evaluating our Music Ontology framework, based on the the data-driven and the task-based evaluation methodologies described in § 4.1.4 and § 4.1.5. We want this evaluation methodology to allow us to validate our ontology with regards to real-world information-seeking behaviours.

We consider evaluating our knowledge representation framework against a dataset of verbalised music-related user needs. We isolate a set of music-related needs drawn by different sets of users, and we measure how well a music information system backed by our knowledge representation frameworks could handle these queries.

Our evaluation methodology involves the following steps, depicted in Figure 4.1.

[1]see `http://trec.nist.gov/` and [Dow06]

1. Constructing a dataset of verbalised user needs;
2. Analysing these needs to extract a set of *features* — recurrent patterns used to describe the information need;
3. Evaluating how well these features map to our knowledge representation framework. The corresponding measure captures the *ontology fit*.

We now detail these three steps of our evaluation process.

### 4.2.1 Step 1 - Constructing a dataset of verbalised user needs

In our selection of user needs, we are not concerned about information retrieval use-cases, but about structured data retrieval [vR99]. We do not consider documents (e.g. a search string or an audio file) as queries with a ranking of relevant documents as answers. In a structured data retrieval scenario, the information needs are expressed by users through a precise question and they expect exact matches to be returned. We therefore consider *queries*, not *search statements* [TS92].

Moreover, we perform a similar evaluation process on several datasets of verbalised user queries. We can distinguish amongst several communities of users, and our Music Ontology framework might perform differently for each of them. We want to compute an ontology fit for each of these communities.

### 4.2.2 Step 2 - Extracting query features

We consider several alternatives for extracting features from our dataset of user needs.

- We can use the results of previous works in extracting query features from similar datasets;
- We can extract features from the dataset by following a statistical approach. We identify recurrent concepts within the dataset;
- We can manually extract features from a random sample of the dataset.

We also consider extracting a *weight* $w_f$ for each feature $f$, capturing the relative importance of $f$ within the dataset. Moreover, these weights are normalised so that their sum is equal to one.

$$w_f = \frac{\text{number of queries that contain the feature } f}{\sum_g \text{number of queries that contain the feature } g} \tag{4.1}$$

### 4.2.3 Step 3 - Computing the ontology fit

The Music Ontology was designed to not duplicate terms that could be borrowed from other web ontologies (for example, `foaf:Person`, `dc:title` or `po:Broadcast`). We take into account this design choice. In the last step of our evaluation process, we therefore consider terms from FOAF, Dublin Core and the Programmes Ontology[2].

We develop an ontology fit measure, capturing how well the extracted features can be mapped to our ontology. For a query feature $f$, we define $\delta$ as follows.

[2]`http://www.bbc.co.uk/ontologies/programmes/`

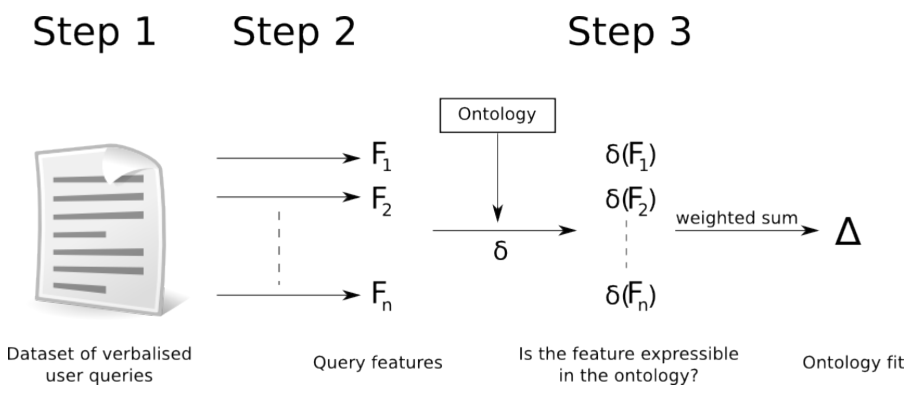


Figure 4.1: Depiction of the different steps involved in our query-driven evaluation

$$\delta(f) = \begin{cases} 1 & f \text{ is expressible within the ontology} \\ 0 & \text{otherwise} \end{cases} \tag{4.2}$$

Our ontology fit measure for a set of verbalised queries $Q$ is then the weighted sum of the $\delta(f)$, for each feature $f$ extracted from $Q$.

$$\Delta = \sum_f w_f * \delta(f) \tag{4.3}$$

The ontology fit measure $\Delta$ therefore captures how possible it is to map a set of user needs to queries expressed within our Music Ontology framework. The closer $\Delta$ is to one, the more our ontology can be used to express the dataset of user queries. We use the ontology fit measure to validate our ontology with regards to real-world user needs.

### 4.2.4 Discussion - query-driven ontology evaluation

This evaluation methodology corresponds to a particular kind of a task-based evaluation methodology, where the task is simply to be able to answer a set of musical queries. The gold-standard associated with this task is that such queries are fully expressed in terms of our knowledge representation framework — the application has a way to derive accurate answers for all of them.

This evaluation methodology also corresponds to a particular kind of data-driven evaluation. We start from a corpus of text, corresponding to our dataset of verbalised user needs, which we analyse and try to map to our knowledge representation framework.

A similar query-driven evaluation of an ontology-based music search system is performed by Baumann et al [BKN02]. They gather a set of 1500 verbalised queries issued to their system, which they cluster in different categories and only analyse qualitatively.

## 4.3 Evaluation of the Music Ontology framework

We want to evaluate our representation framework against a large dataset, holding musical needs drawn by a wide range of users. We consider two main categories of users: music non-experts and users of music libraries. We derive an ontology fit measure for each of these categories.

### 4.3.1 Music non-experts needs

We first consider verbalised queries drawn by music non-experts. We measure how well these queries can be expressed within our Music Ontology framework using our ontology fit measure. We consider the three alternatives mentioned in § 4.2.2 for extracting features from a dataset of verbalised user queries. We use the results of previous works in § 4.3.1.1. Then, we perform a statistical analysis on a large dataset of verbalised user queries in § 4.3.1.2. Finally, we perform a manual analysis on a sample of this large dataset in § 4.3.1.3.

#### 4.3.1.1 Evaluating against previous studies of user needs

We consider evaluating our knowledge representation framework using previous analysis of non-expert user needs. Such analysis leads to the categorisation of the query *type* (e.g. queries aiming at identifying a particular musical item, queries aiming at researching a particular aspect of a musical work, etc.), of the query *context* (the intended use for the requested information) and of the query *features* (recurrent patterns used to describe the information need). We are especially interested in the categorisation of query features, as it leads directly to the results of the second step of our evaluation methodology.

Bainbridge et al. [BCD03] analyse 502 queries gathered from Google Answers[3]. Google Answers allows users to post a particular question, which others can answer. Lee et al. [LDJ07] analyse 566 queries from the same source, restricting themselves to queries aiming at identifying a particular recording or a particular artist. Both extract a set of recurrent features in such queries. The extracted features along with their correspondences to Music Ontology terms are summarised in table 4.1.

The corresponding ontology fit $\Delta$ is therefore 0.749 for Lee's analysis, and 0.958 for Bainbridge's analysis.

Such ontology fit measures are arguable. The proper term to choose within the Music Ontology framework for one of these features is highly dependent on its actual context. It might be the case that one of these features is expressible within our framework in one context, but not in another. For example, for the "related work" feature, "it was a cover of *a*" is expressible, but "it was in the charts at the same time as *a*" is not. These features are too general to provide a good basis for deriving an ontology fit measure.

#### 4.3.1.2 Corpus-driven evaluation

We now perform an evaluation inspired by the data-driven evaluation proposed by Brewster et al. [BADW04] and reviewed in § 4.1.5. We use a statistical analysis on a dataset of user queries to derive information features, and we try to map the results of such an analysis onto Music Ontology terms.

We sample verbalised user needs from both Google Answers and Yahoo Questions[4]. We aggregated the whole Google Answers archive, in the music category (3318 verbalised user needs), and 46% of Yahoo Questions (4805 verbalised user needs).

We use the Porter Stemming algorithm [Por80] to remove common morphological and inflectional endings from words in the dataset, in order to derive the word counts in table 4.2. We also

[3]Google Answers archives are available at `http://answers.google.com/`, as the service is no longer running.
[4]Yahoo Questions is available at `http://answers.yahoo.com/`.

| Features used in queries | Music Ontology term | % of queries containing the feature Bainbridge et al. [BCD03] | Lee et al. [LDJ07] |
|---|---|---|---|
| Lyrics | `mo:Lyrics` | 28.9 | 60.6 |
| Date | `event:time` | 31.9 | 59.2 |
| Media | `mo:Medium` | - | 44.0 |
| Genre | `mo:genre` | 32.7 | 35.5 |
| Uncertainty | - | - | 30.7 |
| Lyrics description | - | - | 30.0 |
| Used in movie/ad | - | - | 30.0 |
| Gender of artist | `foaf:gender` | - | 20.5 |
| Musical style | `event:factor`[a] | - | 19.8 |
| Artist Name | `foaf:name` | 55.0 | 19.3 |
| Orchestration | `mo:Orchestration` | 13.5 | 16.8 |
| Related work | `mo:MusicalWork`[b] | - | 15.9 |
| Lyrics topic | `dc:subject` | 2.6 | 15.4 |
| Where heard | `event:place` | 24.1 | 14.7 |
| Affect/Mood | - | 2.4 | 14.0 |
| Musical Work | `mo:MusicalWork` | 35.6 | 13.6 |
| Used in scene | `mo:Signal` | - | 13.3 |
| Audio/Video example | - | 4.4 | 10.8 |
| Similar | -[c] | 4.6 | 9.2 |
| Tempo | `mo:tempo` | 2.4 | 7.6 |
| Nationality of music/artist | `fb:nationalityNoun` | 12.5 | 4.2 |
| Related event | `event:Event` | - | 4.2 |
| Language | `dc:language` | 2.0 | 3.7 |
| Record | `mo:Record` | 12.2 | 2.7 |
| Melody description | `so:Motif` | - | 0.7 |
| Label | `mo:Label` | 5.4 | 0.1 |
| Link | `foaf:page` | 2.9 | - |

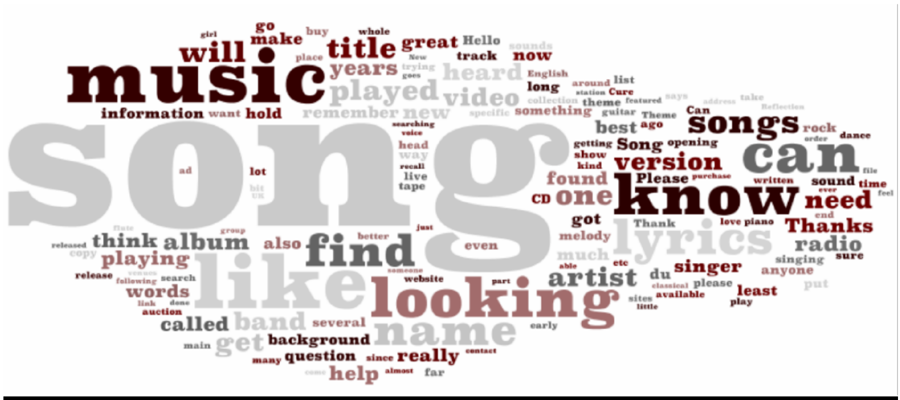
[a]For example, to express that a particular performance has a given stylistic influence, we add this influence as a factor of the performance.

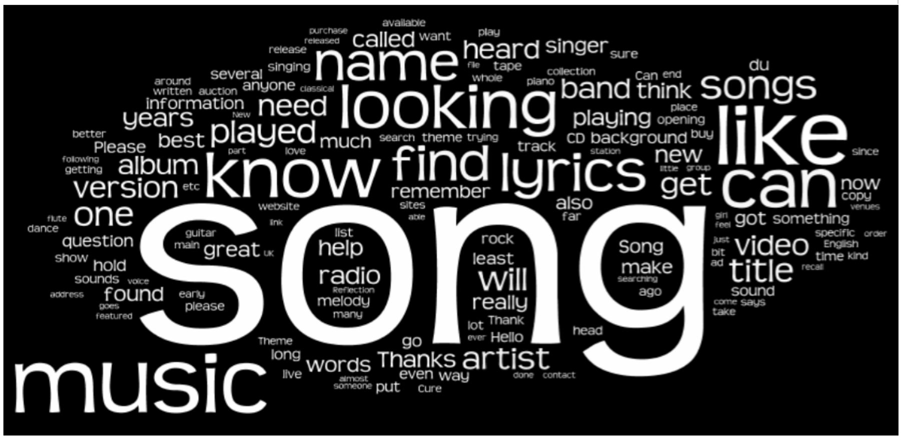
[b]The relationship between the sought work and the related work can be defined using the terms in § 3.3.2.

[c]Such use-cases could be handled using `mo:similar_to`, but it is too vague. We would need more context to choose the appropriate Music Ontology term.

Table 4.1: Comparison of the features identified in [BCD03] and in [LDJ07] along with corresponding Music Ontology terms

Visualisation of term occurrences in
4805 questions gathered from Yahoo Questions





Visualisation of term occurrences in
3318 questions gathered from Google Answers

Figure 4.2: Depiction of predominant terms in our two datasets

mention in this table Music Ontology terms that are related to these frequently occurring terms, therefore giving us a broad idea of the coverage of our ontology. These predominant terms are also depicted in Figure 4.2.

Most user queries include editorial information (artist name, track name, etc.), as spotted in previous analyses of similar datasets. When including some information about a musical item, this information will most likely be related to vocal parts: singer, lyrics, etc. The three musical *genres* cited in this list of predominant terms are "rock", "classical" and "rap". The queries often include information about space and time (e.g. when and where the user heard about that song). They also include information about the access medium: radio, CD, video, online, etc. A large part of the queries include personal feelings, illustrated by the terms "love" or "like". Finally, some of them include information about constituting parts of a particular musical item (e.g. "theme"). We could consider these predominant words as query features and their counts as a weight. However, the same problem as in the ontology fit derived in § 4.3.1.1 also arises. The Music Ontology term corresponding to one of these features is highly context-dependent.

There are two ways to overcome these issues. On the one hand, we can keep our evaluation purely on a *lexical* level. We are particularly interested in such an evaluation because it allows us to include other music-related representation frameworks which are not ontologies but just specifications of data formats, therefore providing some insights for comparison. However, such a lexical comparison needs another methodology than the methodology defined in § 4.2 to derive an ontology fit measure. On the other hand, we can extract underlying topics from our corpus of verbalised user needs, and consider these topics as query features. We therefore move our evaluation to the *conceptual* level.

**Evaluation at the lexical level.** We now derive a measure of the lexical coverage of our ontology. We first produce a vector space representation of these verbalised user needs and of labels and comments within the Music Ontology OWL code. We first remove common stop words. We then map the stemmed terms to vector dimensions, and we simply count the terms in a document to create one vector for our dataset, and one for our ontology.

For example, the two documents "When did the Beatles released the White album" and "They were the members of the Beatles" would be stemmed and translated to the following matrix.

$$\begin{pmatrix} 1 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 & 0 \end{pmatrix} \tag{4.4}$$

with the following mapping of dimensions to stemmed terms:

$$\begin{array}{cccccc} 1 & 2 & 3 & 4 & 5 & 6 \\ \textit{album} & \textit{thei} & \textit{beatl} & \textit{member} & \textit{releas} & \textit{white} \end{array} \tag{4.5}$$

Then, we apply a *tf-idf* [SM83] transform to the resulting matrix — we multiply a term frequency count by an inverse document frequency count, measuring the number of occurrences of a word in the entire corpus. The term-frequency count is defined as follows.

$$\mathsf{TF}(i,j) = \frac{M(i,j)}{\sum_k M(k,j)} \tag{4.6}$$

where $M(i,j)$ is the vector space representation of our documents.

| Term | Count | Related Music Ontology term |
|---|---|---|
| song | $7,789$ | `mo:MusicalWork` |
| music | $2,148$ | - |
| play | $1,691$ | `mo:Performance` |
| lyric | $1,584$ | `mo:Lyrics` |
| sing | $1,489$ | `mo:singer` |
| hear | $1,177$ | `mo:listener` |
| band | $1,037$ | `mo:MusicGroup` |
| i | $1,021$ | `foaf:Person` |
| sound | $1,005$ | `mo:Sound` |
| love | 972 | - |
| say | 833 | `mo:Lyrics` |
| artist | 832 | `mo:MusicArtist` |
| album | 813 | `mo:Record` |
| make | 804 | `foaf:maker` |
| video | 728 | - |
| online | 727 | `foaf:Document` |
| title | 675 | `dc:title` |
| time | 668 | `event:time` |
| call | 656 | `foaf:name` |
| year | 648 | `event:time` |
| version | 632 | `mo:Arrangement` |
| remember | 558 | - |
| rock | 550 | `mo:Genre` |
| girl | 539 | `foaf:gender` |
| listen | 516 | `mo:listener` |
| word | 506 | - |
| radio | 465 | `po:Service` |
| great | 459 | - |
| cd | 448 | `mo:CD` |
| download | 441 | `mo:available_as` |
| feel | 436 | - |
| start | 428 | `tl:start` |
| end | 428 | `tl:end` |
| record | 402 | `mo:Record` |
| guitar | 397 | `mit:Guitar` |
| dance | 389 | - |
| place | 384 | `event:place` |
| theme | 381 | `so:Motif` |
| show | 364 | `po:Programme` |
| before | 361 | `tl:before` |
| long | 359 | - |
| head | 328 | - |
| voice | 326 | `mo:singer` |
| part | 323 | `af:StructuralSegment` |
| track | 318 | `mo:Track` |
| classic | 317 | `mo:Genre` |
| release | 313 | `mo:Release` |
| rap | 312 | `mo:Genre` |

Table 4.2: 48 predominant music-related stemmed terms in the music non-expert datasets and a possible mapping to Music Ontology terms. We omit terms related to the query itself (e.g. “find”, “please”, etc.).

The inverse document frequency count is defined as follows.

$$\mathsf{IDF}(i) = \log \frac{J}{|d_j : M(i,j) > 0|} \tag{4.7}$$

where $J$ is the total number of documents in the corpus, and $|d_j : M(i,j) > 0|$ is the number of documents holding the term corresponding to the dimension $i$.

The tf-idf transform is then given by $\mathsf{TFIDF}(i,j) = \mathsf{TF}(i,j) * \mathsf{IDF}(i)$. For our example document vectors in (4.4) the result of a tf-idf transform is as follows.

$$\begin{pmatrix} 0.17 & 0 & 0 & 0 & 0.17 & 0.17 \\ 0 & 0.23 & 0.23 & 0.23 & 0 & 0 \end{pmatrix} \tag{4.8}$$

We finally compute a cosine distance between the vector corresponding to our dataset of user queries and the vector corresponding to our Music Ontology framework, to capture the inter-document similarity. The cosine distance between two vectors $A$ and $B$ is defined as follows.

$$\mathsf{cosine}(A,B) = \frac{A * B}{||A|| * ||B||} \tag{4.9}$$

We also include in our vector space other related representation frameworks, which we also compare to the dataset of user queries to derive the results in table 4.3.

We first note that the results in this table are not comparable with the ontology fit results derived in the rest of this chapter. They are not computed using the methodology described in § 4.2. We also note that our Music Ontology framework performs better than the other representation framework — it is closer to the dataset of user queries. These results are due to the fact that our ontology encompasses a wider scope of music-related information than the others, which are dedicated to specific use-cases. For example, XSPF is dedicated to represent playlist, iTunes XML and hAudio to simple editorial metadata, Variations3 to music libraries and AceXML to content-based analysis and machine learning. Most of these frameworks have anchor points in the Music Ontology, as illustrated in § 6.3. The lexical coverage of the Music Ontology framework is therefore higher. Of course, this measure is very crude, and just captures the lexical overlap between specification documents and our dataset of user queries. It can serve for comparison purposes, but not to validate our framework against this dataset.

**Evaluation at the conceptual level.** We now want to go beyond this lexical layer. We try to extract from our dataset a set of underlying *topics*, which constitute our query features. We then consider these topics as our query features, and compute an ontology fit measure from them by following the methodology described in § 4.2.

We consider that our corpus of musical queries reflects the underlying set of topics it addresses. A common way of modelling the contribution of these topics to the $i$th word in a given document (in our case, a musical query) is as follows.

$$P(w_i) = \sum_{j=1}^{T} P(w_i|z_i = j).P(z_i = j) \tag{4.10}$$

where $T$ is the number of latent topics, $z_i$ is a latent variable indicating the topic from which the $i$th word was drawn, $P(w_i|z_i = j)$ is the probability of the word $w_i$ under the $j$th topic, and $P(z_i = j)$ is the probability of choosing a word from the $j$th topic in the current document. For

| Ontology | Similarity |
|---|---|
| Music Ontology | 0.0812 |
| ID3 version 2.3.0 | 0.0526 |
| hAudio | 0.0375 |
| Musicbrainz | 0.0318 |
| XSPF | 0.026 |
| ACE XML | 0.0208 |
| iTunes XML | 0.0182 |
| ID3 version 2.4.0 | 0.0156 |
| Variations3 FRBR-based model, phase 2 | 0.0112 |
| FRBR Core & Extended | 0.0111 |
| MODS | 0.0055 |
| MPEG-7 Audio | 0.0013 |

Table 4.3: Cosine similarities between vectorised specification documents and the music non-experts dataset. We use labels and descriptions of terms for web ontologies, and textual specifications for other frameworks

example, in a corpus dealing with performances and recording devices, $P(w|z)$ would capture the content of the topic. The performance topic would give high probability to words like venue, performer or orchestra, whereas the recording device topic would give high probability to words like microphone, converter or signal. Whether a particular document concerns performances, recording devices or both would be captured by $P(z)$

The Latent Dirichlet Allocation (LDA [BNJ03]) provides such a model. In LDA, documents are generated by first picking a distribution over topics from a Dirichlet distribution, which determines $P(z)$. We then pick a topic from this distribution and a word from that topic according to $P(w|z)$ to generate the words in the document. We use the same methodology as in [GS04] to discover topics.

We use an approximate inference algorithm via Gibbs sampling for LDA [Hei08][5]. We first pre-process our dataset of musical queries by stemming terms, removing standard "stop" words used in computational linguistics and removing words that appear in less than five queries. Repeated experiments for different number of topics (20, 50, 100 and 200) suggested that a model incorporating 50 topics best captures our data. We reach the set of topics illustrated in table 4.4.

We consider these topics as our query features. For each topic, we sum the number of words that has been assigned to it in the training data. We normalise the resulting numbers to derive a weight for each of our query features. We try to manually map each topic to terms within our Music Ontology framework to derive the ontology fit measure described in § 4.2.3. The corresponding ontology fit measure is 0.723.

However, this measure of the ontology fit is still arguable. Some of the topics inferred are not sufficiently precise to be easily mapped to Music Ontology terms. A subjective mapping still needs to be done to relate the outputted topics with a set of ontology terms. Moreover, some crucial query features are not captured within the outputted topics. For example, a lot of queries include an implicit notion of uncertainty (such as "I think the title was something like Walk Away"), which is not expressible within our ontology.

A promising approach is to use a Correlated Topic Model [BL07], which also models the relationships between topics. This would allow us to not only evaluate the coverage of concepts within our ontology, but also the coverage of the relationships between these concepts. It remains

[5] We use the implementation available at `http://gibbslda.sourceforge.net/`

| Topic 4 | Topic 13 | Topic 19 | Topic 23 | Topic 48 | Topic 49 |
|---|---|---|---|---|---|
| band | album | song | music | play | live |
| rock | track | singer | piece | piano | concert |
| metal | artist | female | classical | note | perform |
| member | cover | sung | sheet | chord | tour |
| punk | release | male | composition | key | date |
| drummer | title | lead | instrument | tuning | ticket |
| guitarist | name | vocalist | score | scale | opera |
| quit | purchase | chorus | piano | melody | stage |
| beatles | bought | artist | orchestra | major | show |
| zeppelin | obtain | sound | choral | minor | play |

Table 4.4: Top words in sample topics inferred through Latent Dirichlet Allocation over our dataset of musical queries

future work to develop an accurate measure for evaluating how the inferred graph of topics can be mapped to an ontological framework.

#### 4.3.1.3 Manual Evaluation of the ontology fit

We now want to derive a more accurate ontology fit measure. In order to do so, we manually extract the underlying logical structure of these queries and see how well these logical structures can be expressed within our representation framework.

We here consider a random sample of 40 queries drawn from the dataset of user needs used in § 4.3.1.2, corresponding to 0.5% of the queries available within the Google Answers archive.

We manually pre-process every verbalised query $q$ in this sample to extract a set $\alpha(q)$ of query features. These features are logical sentences encoding the query. For example, a query holding the sentence "the performer of the song is Chet Baker" would lead to the following two features:

$$\alpha(q) = \{\mathsf{performer}(S, P), \mathsf{name}(P, \text{'Chet Baker'})\}$$

We do not follow exactly the manual data analysis methodology used by Lee et al. [LDJ07], which partially structures the original queries by delimiting the parts that correspond to a particular recurrent feature. Indeed, it is important for our purpose that we extract the whole logical structure of the query. This will lead to a more accurate ontology fit measure (but derived from a smaller dataset) than in the previous sections.

Once these queries have been pre-processed, we assign a weight for each distinct feature. Such weights are computed as described in § 4.2.2. We give the main query features, as well as the corresponding weight and the corresponding Music Ontology term, in table 4.5. We then compute our ontology fit measure, as described in § 4.2.3. We find an ontology fit measure of 0.714.

#### 4.3.1.4 Discussion - music non-experts evaluation

The different variants of our query-driven evaluation made in this section all lead to a similar ontology fit measure. Around 70% of the information held within a dataset of music non-experts queries is expressible within our Music Ontology framework.

Over the different evaluations made in this section, we found that the main features that are not expressible within our framework are the following.

- Uncertainty - e.g. "I don't remember if the song had drums in it";

| Feature | Weight | Corresponding term |
|---|---|---|
| title(Item, Title) | 0.085 | `dc:title` |
| maker(Item, Maker) | 0.058 | `foaf:maker` |
| lyrics(Item, Lyrics) | 0.054 | `mo:Lyrics` |
| time(Item, Time) | 0.042 | `dc:date` |
| uncertain(Statement) | 0.042 | - |
| similar(Item1, Item2) | 0.035 | - |
| based_near(Person, Place) | 0.035 | `foaf:based_near` |
| place(Event, Place) | 0.031 | `event:place` |

Table 4.5: Predominant query features in a random sample of the Google Answers dataset, along with their weights and corresponding terms in the Music Ontology framework

- Partial characterisation of the lyrics - e.g. "One part of the lyrics was 'put your hands up' "[6];
- Emotions related to the music itself - e.g. "This song was really sad";
- Description of related media - e.g. "In the music video, the band was playing in a forest and the singer was trapped under ice" or "The artist was on the cover of that magazine";
- Other cultural aspects, such as the position of a track in the charts.

Future work on the Music Ontology should therefore focus on these points.

### 4.3.2 Users of music libraries

Gathering a dataset of music library user needs is difficult. We therefore adapt our approach to cope with the lack of publicly available datasets.

#### 4.3.2.1 Methodology

Saxton and Richardson [SR02] present an evaluation methodology for reference services in libraries based on the sampling of real-world questions and on the evaluation of the corresponding transactions on a number of dimensions, including completeness, usefulness, user satisfaction and accuracy. Sugimoto [Sug07] isolates such reference questions in order to evaluate the performance of music libraries. He then analyses the corresponding transactions for a number of music libraries in the United States.

We evaluate our representation framework using a similar methodology. We consider a reference set of queries covering a wide range of possible query types. We then manually extract query features, and follow the process described in § 4.2 to derive an ontology fit measure. We therefore evaluate the performance of the ontology by quantifying how an ontology-backed system would perform if it were occupying the role of the librarian. Such a methodology is similar to the methodology we adopted in § 4.3.1.3, except that we filter the dataset to leave a small sample of representative questions prior to the actual evaluation, instead of using a random sample of questions. The accuracy of such an evaluation is arguable, as it does not include information about the predominance of a query feature in a real-world dataset. However, it gives us a measure of how well a representative set of query features is covered by our ontoloy.

[6]This particular point is certainly made really important by the bias our dataset has towards english-speaking users. For example, Baumann [Bau05] reports the case of an user stating "I am not interested in lyrics in general, because my English is too bad to understand something".

| Feature | Weight | Corresponding term |
|---|---|---|
| composer(Item, Person) | 0.158 | `mo:composer` |
| factor(Work, Factor) | 0.105 | `event:factor` |
| title(Item, Title) | 0.105 | `dc:title` |
| available_as(Item, Location) | 0.105 | `mo:available_as` |
| address(Item, Address) | 0.053 | `vCard:adr` |
| city(Location, City) | 0.053 | `vCard:locality` |
| instrumentation(Item, Instrumentation) | 0.053 | `mo:instrument` |
| made_in(Event, Place) | 0.053 | - |
| print(Item) | 0.053 | - |
| poster(Item) | 0.053 | - |
| wood_print(Item) | 0.053 | - |
| about(Person, About) | 0.053 | -[7] |
| performer(Item, Person) | 0.053 | `mo:performer` |
| lyrics(Item, Lyrics) | 0.053 | `mo:Lyrics` |

Table 4.6: Predominant query features in a Sugimoto's queries, along with associated weights and terms in the Music Ontology framework

#### 4.3.2.2 Dataset of user queries

We consider re-using the questions selected in Sugimoto's study [Sug07] from a binder of recorded reference questions asked at the University of North Carolina Chapel Hill Music Library between July 15, 1996 and September 22, 1998. These questions were chosen to cover a typical range of possible questions asked in a music library. These questions are:

1. What is the address for the Bartok Archive in NY?
2. Can you help me locate Civil War flute music?
3. I am a percussion student studying the piece "Fantasy on Japanese Wood Prints" by Alan Hovhaness. I wondered if there was any information available about the actual Japanese wood prints that inspired the composer. If so, what are their titles, and is it possible to find prints or posters for them?
4. Do you have any information on Francis Hopkinson (as a composer)?
5. What are the lyrics to "Who will Answer"? Also, who wrote this and who performed it?

#### 4.3.2.3 Ontology fit

We start by extracting features from these five queries as in § 4.3.1.3. We reach the query features, associated weights and Music Ontology terms in table 4.6. This table suggests an ontology fit measure of 0.789. Our ontology therefore performs slightly better for this particular dataset than for the music non-experts dataset. These results can be explained by the diversity of the queries drawn by music non-experts. For example, one query analysed within § 4.3.1.3 describes in great levels of detail a music video in order to get to the name of a track. Such descriptions are not expressible within our framework and make the ontology fit drop.

## 4.4 Summary and discussion

In this chapter, we devised a query-driven evaluation process for music ontologies, based on the data-driven and task-based ontology evaluation methodologies. We create a dataset of user

| Dataset | Evaluation method | Section | Ontology fit ($\Delta$) |
|---|---|---|---|
| Casual users | Using results of previous analysis | § 4.3.1.1 | 0.749 for [LDJ07]<br>0.958 for [BCD03] |
| | Statistical analysis | § 4.3.1.2 | 0.723 |
| | Manual analysis | § 4.3.1.3 | 0.714 |
| Music library users | Manual analysis | § 4.3.2.3 | 0.789 |

Table 4.7: Summary of the ontology fit results described in this chapter

queries and measure how well these queries fit within our knowledge representation framework. We end up quantifying how well a system based on our representation framework could help answering these queries.

A number of alternatives can be used for each step of such an evaluation process. First, there are several categories of users, which are interesting to handle separately, as our ontology may perform differently for each. Then, there are several ways of performing an analysis of user queries. We summarise in table 4.7 the results obtained in this chapter, investigating different alternatives for each of these steps. Our ontology covers more than 70% of the different datasets considered. We identified the main features that are lacking from our ontology in § 4.3.1.4.

The flexibility of our ontology is also highlighted in another evaluation of music representation systems [CGVD08], identifying a number of music-related information clusters and determining whether the evaluated representation systems can handle them. Our Music Ontology is reported to handle all of them but one: the ability of expressing meta-information e.g. "*a* was conducted by *b*" is believed by *c*', although we can argue that such information is handled by the different reification mechanisms mentioned in § 2.2.4.2 and § 2.2.4.5.

We also performed a lexical comparison of different music representation framework. This comparison captured how lexically close a particular representation framework is from a dataset of music non-experts queries. We found that our Music Ontology framework performs sensibly better than the others.

All the results described in this chapter evaluate a particular characteristic of our ontology: its coverage of real-world user needs. However, a number of other characteristics would be interesting to capture as well. For example, we might want to evaluate the *verbosity* of the ontology – how many ontology terms are needed to express a particular information feature. We might also want to evaluate the *popularity* of the ontology – how many documents reusing Music Ontology terms are available on the Web. It remains future work to evaluate fully such characteristics, although some insights about the popularity of the Music Ontology framework will be given in chapter 6.

# Chapter 5

# Music processing workflows on the Web

In chapter 3, we described the Music Ontology, able to express a range of music-related information, from purely editorial to complex temporal annotations. We evaluated how wide this range is in chapter 4. The Music Ontology framework provides a set of web identifiers and corresponding structured representations for a number of music-related concepts and relationships. Now, structured music-related data can be published on the Web and linked to these identifiers, which a user agent can follow in order to get to the actual meaning of such data.

Such data can be pre-existing—it is stored somewhere and looked up when requested. It can also be derived from the multimedia content, automatically. In the case of musical audio, one can automatically derive keys, rhythm, timbre, melody, etc, as reviewed in Appendix A. Other data can be derived from symbolic notations, speech audio, images or videos.

Automatically derived data is already useful on its own to perform content-based information retrieval tasks (e.g. 'Give me tracks where the melodic content is similar to this melody') [YI99], but could be even more useful if it were interlinked with other data (e.g. 'Give me performances of that Jazz piece where the piano solo includes this musical phrase'). The Music Ontology framework allows us to express such relationships.

All these different kinds of web data, originating from online data sources or from analysis tools, can be combined to produce further information. For example, as pointed out by Garcia and Celma [GC05], we might want to specify that:

> A "danceable" track is a track with an average tempo of around 120 beats per minute, a high loudness, and which is in the playlist of a club disc jockey.

Such a *rule* gathers two statements that can be automatically derived from the content, involving loudness and tempo detection, and playlisting information. This rule can also be viewed as a workflow, combining a loudness and a tempo extraction tool with other information. Information, regardless of how it was generated, needs to be available in a common form to allow such rules to be formulated and automatically interpretable. Moreover, it needs to be interlinked across the different data sources involved. In our example, we need to know that the tracks we analysed to get to the tempo and the loudness values are indeed the tracks involved in the disc jockey's playlist. Using web identifiers for cross-source identity[1], RDF for automatically processable information

[1]along with `owl:sameAs` statements, which are used to state that two identifiers identify the same thing, and

and logical rules for deriving new information provides a good ground to handle such hybrid deductions.

We describe here a framework supporting such use cases. We define a knowledge representation framework supporting such rules, combining information and analysis tools. The information derived from such a combination can be given semantic value by being linked to terms within an ontology, such as the Music Ontology framework described in chapter 3. We define a publication mechanism, for making the corresponding results available on the Web. Our final aim is to create autonomous music-related 'Semantic Web agents' (as defined by Berners-Lee et al. [BLHL01]), processing machine-readable web content and musical items to publish new information. This new information can then be reused by further agents, or for a wide range of purpose, including personal collection management, search , visualisation and recommendation. We will propose an implementation of such a Semantic Web agent in chapter 7 and such applications in chapter 8. These agents constitute the dynamic part of our framework.

We first give an overview of previous work in multimedia information systems combining information from multiple origins in § 5.1. We provide a knowledge representation framework for combining multimedia processing tasks and other information available on the Semantic Web in § 5.2. We describe two different mechanisms for on-demand publishing of information derived from such a combination in § 5.3. Finally, we compare our overall framework to related work in § 5.4.

## 5.1 State of the art

Our review of the state of the art is divided in five main categories. We focus in § 5.1.1 on previous work on hybrid multimedia information systems, handling both content-based and external information. We then focus in § 5.1.2 on the management of music processing *workflows*, defining how different analysis processes can be combined to derive new information. We review different distributed music information systems in § 5.1.3. We then focus on logical frameworks for combining information from multiple origins to derive higher-level information in § 5.1.4. Finally, we overview frameworks for keeping track of the process leading to such derived information in § 5.1.5. In each of these categories, we isolate requirements that will guide the design of our framework for music processing workflows, described in § 5.2 and § 5.3.

### 5.1.1 Designing hybrid Multimedia information systems

The design of multimedia information systems handling both information derived from the content ('this image has the following colour histogram') and external information ('this is a picture of Tom') has been extensively researched over the last years. The amount of available multimedia information motivates the use of highly efficient models for managing both kinds of information.

Grosky [Gro94] describes the different parts of a model backing a multimedia information system. A model for multimedia objects needs to capture the actual multimedia objects, data derived from the content ('features'), and domain-specific information. Then, information held within these three parts can be combined to derive higher-level information, as highlighted in the "danceable" example in the introduction of this chapter.

inverse functional properties, which imply that two subjects are the same when they share a common object

Grosky [Gro97] also describes some features that are necessary to add to traditional relational database systems [Cod70] to handle the peculiarities of multimedia data. He highlights the need for content-based similarity within such systems, through a set of *functions* that can be embedded within relational queries. A typical use-case of such a mechanism is the following query:

> "Retrieve all video shots showing my friend Tom dancing, given a photograph of Tom."

Some similar systems have been developed in the music domain. Baumann et al. describe an ontology-based framework [BKN02] for integrating music-related information: ID3 meta-data (see § 3.1.2.2) and the results of automatic audio analysis and classification processes. The underlying ontology is mainly focused around tracks, musical features of tracks, artists, and musical genres. The MTG-DB [CKF$^+$04] gathers in a common centralised repository audio content, corresponding meta-data, taxonomies (for example, an instrument classification scheme) and algorithms (for example, an instrument classifier). The system also includes an interface, WaveSurfer, to interact with the items held within the repository. The underlying database schema encompasses audio files, structural segmentations, tracks, artists, albums and categories. The data models used in these two systems are less expressive than the model we developed in chapter 3.

Pachet [PBAB05] describes another music information management system. The data model used is similar to the RDF model described in § 2.2.4.2 and therefore borrows its expressiveness. Some fields[2] in this model may be *computable*. Their value is the output of an automatic extractor. They can either be computed when their value is requested or in batch mode.

These different systems are centralised. All the data and the content-based analysis algorithms are held in a single place. There is no way to make one of these data management systems benefit from the data another may hold. Also, such centralisation makes it difficult to issue federated queries, involving data held (or computable) by all of these systems. Another related shortcoming is the inability to easily define new ways of computing a result by combining data held by different systems. We want our framework to be completely distributed, allowing us to reuse analysis tools and data available in multiple places.

### 5.1.2 Worflow specification for music analysis

A workflow defines how different processes can interact with each other in order to achieve a particular goal. Processing multimedia data often involves complex parallel workflows, composed of many steps, that are likely to be computation-intensive. Special care must therefore be taken in specifying and managing such workflows.

Casati [CCPP95] specifies an ontology of workflows, in a context of integration between database and process modelling technologies. This ontology defines a workflow *schema*, describing the relationships amongst the different tasks constituting the workflow as well as more information about these tasks (preconditions and operations they imply on a persistent database, for example). Then, a workflow *instance* is a particular execution of such a schema. The ontology also defines a number of tasks representing elementary operations (fork, join) for concurrent workflows. Different workflows communicate between themselves through a single persistent database.

Graphical programming languages such as D2K (data-to-knowledge) and its music counterpart M2K (music-to-knowledge [DET05]), a music information retrieval prototyping and evaluation

[2]A field here corresponds to the object of a particular RDF statement.

system, enable the graphical specification of workflows operating over multimedia data. Elementary operations are visualised as nodes in a graph, with inbound and outbound links representing what inputs and parameters they use, and what to do with their outputs. These applications also support the execution of such workflows.

However, these systems are still centralised. It is difficult within these applications to reuse a workflow or a task defined (and perhaps executed) elsewhere. It is also difficult for these applications to use data available remotely. Moreover, these applications do not deal with the related data management issues. For example, it is impossible to persist a result derived from such a workflow, to be able to retrieve it later on. It is also impossible to reverse-engineer a result, to get back to what has been achieved to derive it. We want our framework to tackle these data management issues, and to consider workflows with different tasks or sub-workflows defined and executed elsewhere.

### 5.1.3 Music analysis in distributed environments

In order to overcome the centralisation issues mentioned in § 5.1.1 and in § 5.1.2, recent efforts tried to make different multimedia processing components remotely accessible. Each step in a multimedia analysis workflow can then be remotely executed.

The Marsyas software framework allows the distribution of its music processing components [BT05]. Each of them can be hosted on a different physical host. These components interact with each other using a dataflow network.

The Echonest Analyze API [ech] provides a Web Service returning a set of content-based features dealing with timbre, pitch, rhythm and structural segments from audio files. TU-Vienna provides a set of Web Services[3] returning rhythm and timbre features features from audio files. It also provides a Web Service for the creation and the training of Self-Organising Maps. The MIREX "Do-It-Yourself" Web Service [EDJ07] enables the remote execution of Music Information Retrieval evaluation tasks.

The OMEN (On-demand Metadata Extraction Framework) framework [MMF06] wraps feature extraction and collection management tools in a set of Web Services that are coordinated through a central service. In this system, only lossy representations of the audio content are exchanged amongst the different components. It is impossible to re-create the audio content from these representations. One can ask this system to compute a set of features on a particular music collection, and retrieve the results. However, the OMEN framework does not provide in itself a way to go beyond a first level of representation.

In order to tackle more complex workflows, several frameworks allow us to specify how different tasks backed by Web Services can be orchestrated. The de facto standard for Web Services orchestration is the Business Process Execution Language [bpe]. BPEL workflows can either be *abstract* or *concrete* [YB05]. In the former case, no particular computational or data providing resource is specified for the execution of the different tasks. In the latter case, tasks in the workflow are bound to specific resources. Other workflow languages encompass the interaction between Web Services and a shared database, such as xWFL [YB04] and its 'tuple space' used by processes to communicate with each other. However, most of these workflow languages have no clear operational semantics. Kiepuszewski [Kie03] evaluated different workflow management

[3]available at `http://www.ifs.tuwien.ac.at/mir/webservice/`

products and showed that almost all of them took a different view on the semantics of the workflow constructs.

Our framework for expressing music analysis workflows must tackle this issue by having clear semantics. Moreover, we feel that instead of combining processes to achieve a particular result, it would be more intuitive to directly combine information. For example, when specifying a new music classifier working on Mel-Frequency Cepstrum Coefficients (MFCCs [RJ93]), we just need to specify that we are working on MFCCs. The component interpreting this statement is then free to meet this requirement with suitable information from any provider. The “danceable” example in the introduction of this chapter illustrates just that. We expressed that we need a track to have a particular tempo and a particular loudness to be danceable. This tempo and this loudness can be gathered from manual annotations, or from content-based analysis processes, but that does not change our rule. Our framework will therefore adopt an information-centric approach, where we focus on combining information, not services.

### 5.1.4 Reasoning in Multimedia Information systems

A number of different approaches move towards such a unified way of combining content-based analysis and other information, by wrapping everything into logical queries.

Brink et al. [BMS95] specify a logic-based query language operating over a set of multimedia objects (images or sampled signals) and a number of relations mapping such objects to particular features, along with a confidence level (*‘the certainty of `John Doe` appearing in that picture is of 70%’*). The overall framework allows one to wrap, within a single query, look-ups in several knowledge bases and requests for automatically extracted features.

The work of Hunter and Little [HL05b] provides a way to reason over multimedia-related information expressed within a common representation framework based on Semantic Web technologies. It also provides a way to express inference rules operating over such information using a rule markup language (RuleML) and a mathematical markup language (MathML). They give the following example, expressed within RuleML and an MPEG-7 ontological framework they develop. This rule states that an image region depicts a *Catalyst*, as defined in the FUSION project fuel cell ontology [fus03], if some content-based conditions are filled in.

$$\begin{aligned}\mathsf{DominantColor}(R, C) \wedge C > (112, 112, 112) \wedge \mathsf{Shape}(R, \mathsf{circle}) \wedge \mathsf{magnification}(R, M)\\ \wedge M \geq 10000 \supset \mathsf{depicts}(R, \mathsf{catalyst})\end{aligned} \tag{5.1}$$

Our work extends this framework by also providing the ability to sequentially or concurrently combine logical predicates that correspond to analysis processes, in order to produce predicates such as $\mathsf{Shape}(R, \mathsf{circle})$ in (5.1). We therefore provide the link between the workflow-based and the rule-based approaches. In the context of our “danceable” example in the introduction of this chapter, this would translate to providing a way to answer the following questions. What should we do to get to an average tempo? What are the different possible ways to derive such data? This will lead us towards a knowledge representation framework for combining content-based processes and other information.

### 5.1.5 Proof and provenance mark-up

When automatically deriving new information through a reasoning process as in § 5.1.4, we need to keep track of this process [Siz07]. If we are able to derive numerous similarity statements between a pair of audio signals, we need to know how they were derived. Is it inferred from the fact that they were both quoted in a single playlist or compilation? Are they acoustically similar? Are they structurally similar?

Such tracking can be done using the Proof Markup Language [dSMF06], developed within the Inference Web project [MdS04]. Using PML, we can relate a conclusion, the corresponding premises and the axiom used. Then, it is possible to provide an explanation for every inferred statement.

A similar mechanism to track the provenance of derived data is implemented in a Life Science context, as described by Zhao et al. [ZWG+04]. The provenance tracking is achieved using an RDF schema linking different processes, analysed resources and other derivation information together.

In a more generic Semantic Web context, the different reification approaches mentioned in § 2.2 can also be used to keep track of RDF statements' provenance and other meta-knowledge [SSST08], e.g. origin, confidence, reasoner used, workflow used, derivation time. Our framework also includes provenance tracking capabilities, as described in § 5.2.6.

## 5.2 Modelling Music processing workflows

We now describe our knowledge representation framework (N3-Tr) for combining structured web data and music analysis workflows to derive new information, based on the requirements we drew in § 5.1. We summarise these requirements as follows.

- Our framework must be decentralised. A workflow can use data, tasks or sub-workflows available in different places ;
- Our representation framework must tackle data management issues, including persistence and provenance tracking ;
- Our representation framework must be information-centric. We focus on combining information, not services ;
- Our representation framework must have clear semantics, and explicitly deal with side-effects;
- Our representation framework must provide the ability to relate the derived information to web ontologies, in order to be given semantic value.

We first describe our model for a single processing step in § 5.2.1. We overview the logical framework underlying N3-Tr in § 5.2.2. We describe our N3-Tr representation framework in § 5.2.3. We give some examples of N3-Tr workflows in § 5.2.4. We highlight the decentralised aspects of our framework in § 5.2.5. Finally, we detail how our framework tackles provenance tracking issues in § 5.2.6.

### 5.2.1 Modelling a single processing step

When working on a set of algorithms, the resulting data is often managed as a dictionary of key-value pairs—this may take form of named variables in a Matlab workspace, files in a directory, or files in a directory tree (in which case the keys would have a hierarchical structure). This can lead to a situation in which, after a Matlab session for example, one is left with a workspace full of objects but no idea how each object was computed, other than, perhaps, cryptic clues in the form of the variable names one has chosen.

The semantic content of the data resulting from a set of algorithms is intimately tied to knowledge about the algorithm itself, its inputs and parameters. To capture the semantics of the spectrogram transform described in § $A.2$, we would need to express that 'data object $x$ (usually a 2-dimensional array of real numbers) was computed from signal $y$ using spectrogram algorithm $z$ with parameters $\theta$'.

For example, one way of expressing a spectrogram derived from an audio signal identified by `:signal` is as follows[4].

```
(:signal sig:hann "256" "128") sig:spectrogram :output.
```

The property `sig:spectrogram` links the inputs (audio signal, window function, window size and hop size) to the outputted spectrogram.

Such predicates can be considered as 'built-in' predicates within a particular RDF store implementation offering a querying facility. When queried in the right mode (i.e. with all input arguments and parameters bound), a computation is launched to bind the output arguments to the corresponding results. We then blur the distinction between a 'property' in RDF and a 'function' provided by a software library.

### 5.2.2 Transaction Logic

In order to specify, analyse, schedule and execute complex workflows (such as a serial or a parallel workflow involving several content analysis tools) within our knowledge representation framework, we consider the use of Concurrent Transaction Logic ($\mathcal{CTR}$ [BK96]). We first describe sequential Transaction Logic ($\mathcal{TR}$ [BK93, BK94, BK95]) in § 5.2.2.1. We then describe $\mathcal{CTR}$ in § 5.2.2.2. We give in § 5.2.2.3 examples of constraints expressed in $\mathcal{CTR}$. Finally, we define $\mathcal{CTR}$ rules in § 5.2.2.4.

#### 5.2.2.1 Sequential Transaction Logic

$\mathcal{TR}$ extends the First-Order Logic reviewed in § 2.1.2 with the $\otimes$ connective, where $\phi \otimes \psi$ denotes the composite transaction consisting of transaction $\phi$ followed by transaction $\psi$[5].

The model-theoretic semantics of $\mathcal{TR}$ is based on sequences of database states, called *paths*. Unlike in classical logic where a formula is true over a particular state, the truth of a $\mathcal{TR}$ formula $\phi$ is determined over a finite sequence $< s_1, \ldots, s_n >$ of such states: a path. If a formula $\phi$ is true over $< s_1, \ldots, s_n >$, it means that executing $\phi$ at the state $s_1$ will lead us to the state $s_n$. (5.2) denotes such an *executional entailment*, captured by the operator $\models$, where $P$ is the set of available transaction definitions — the *transaction base*.

[4]In all this chapter, we use the N3 notation reviewed in § 2.2.4.5.

[5]In this chapter, we use Greek letters for identifying transaction logic formulæ.

$$P, s_1, \ldots, s_n \models \phi \tag{5.2}$$

Then, the serial conjunction $\otimes$ is defined as follows.

$$P, s_1, \ldots, s_i, \ldots, s_n \models \phi \otimes \psi \tag{5.3}$$

where

$$\begin{gathered} P, s_1, \ldots, s_i \models \phi \\ P, s_i, \ldots, s_n \models \psi \end{gathered}$$

$\phi$ brings the database from the state $s_1$ to the state $s_i$, and $\psi$ brings the database from the state $s_i$ to the state $s_n$.

Elementary database operations (corresponding to a sequence of database states $< s_i, s_{i+1} >$) are defined by a pair of *oracles*. The *data oracle* $\mathcal{O}^d$ is a mapping from a database state to a set of first-order formulæ. The *transition oracle* $\mathcal{O}^t$ is a mapping from pairs of states to the corresponding elementary updates. Any database and transition semantics can be "plugged into" $\mathcal{TR}$ by defining new oracles. Defining our knowledge representation framework will require the creation of such oracles.

#### 5.2.2.2 Concurrent Transaction Logic

$\mathcal{CTR}$ has the following new connectives and operators, which we describe by their intuitive procedural semantics for now.

- $\phi \mid \psi$ means 'Execute $\phi$ and $\psi$ concurrently, and commit the database changes corresponding to $\phi$ and $\psi$ iff both $\phi$ and $\psi$ succeed';
- $\odot\phi$ means 'Execute $\phi$ in isolation, and commit the database changes corresponding to $\phi$ iff $\phi$ succeeds'.

The model-theoretic semantics of $\mathcal{CTR}$ is based on the notion of *multi-paths*. A multi-path is a finite sequence of paths, where each path correspond to a period of continuous execution, separated by periods of suspended execution during which other processes may execute. For example, if the multi-path $< s_1 s_2 s_3, s_4 s_5 s_6 >$ represents the execution history of the $\mathcal{CTR}$ formula $\phi$ , it means that in the first period, $\phi$ changes the database from the state $s_1$ to the state $s_3$ through an intermediary step $s_2$. $\phi$ is then suspended until the database is at the state $s_4$. When $s_4$ is reached (due to other processes), $\phi$ brings the database to the state $s_6$. Multi-paths support an incomplete system of transactions. A complete system of transactions is supported by a simple path, as defined in § 5.2.2.1.

For example, if the $\mathcal{CTR}$ formula $\psi$ is entailed on the multi-path $< s_3 s_4, s_6 s_7 >$, then $\phi \mid \psi$ is entailed on the path $< s_1 s_2 s_3 s_4 s_5 s_6 s_7 >$. $\phi$ first makes the database evolve from $s_1$ to $s_3$. Then, $\psi$ makes the database go from $s_3$ to $s_4$. Then, $\phi$ makes the database go from $s_4$ to $s_6$. Finally, $\psi$ leads us to the state $s_7$. If $\xi$ is entailed on the multi-path $< s_6 s_7, s_8 s_9 >$, then $\phi \otimes \xi$ is entailed on $< s_1 s_2 s_3, s_4 s_5 s_6 s_7, s_8 s_9 >$.

An isolated transaction, defined using the operator $\odot$, is entailed on a path, not a multi-path — the process is not interleaved with other processes.

$\mathcal{TR}$ and $\mathcal{CTR}$ are conservative extensions of FOL. They reduce to FOL for formulæ that do not involve state transitions, but just query the current state. If $P$ and $\phi$ are both first-order formulæ, then

$$P, D \models \phi \text{ iff } P \wedge D \models^c \phi \tag{5.4}$$

where $\models^c$ denotes entailment in FOL and $\models$ denotes executional entailment.

The ability of $\mathcal{CTR}$ to model and schedule workflows is discussed in [DKRR98], and an application in Bioinformatics, a large scale genome-mapping project in a real research environment, is described in [BSR96].

#### 5.2.2.3 Constraints

Classical conjunction can be used alongside the $\mathcal{CTR}$ connectives to express constraints, such as pre-conditions and post-conditions. Pre-conditions are especially useful for describing constraints over the inputs of a process.

**Example 1.** We can express that a beat tracking algorithm [DP07] requires an onset detection function[6] derived from an audio signal as an input.

$$\text{:onset\_detection\_function}(\text{:signal}, F) \otimes \text{:beat\_indices}(F, I) \tag{5.5}$$

**Example 2.** Another example of pre-conditions is to restrict the modifications we can bring to the database. For example, we may express that an average tempo can be inserted in a database only if it is derived from a track involving drums. We express this pre-condition using the terms defined in § 3.3.2.2.

$$(\text{mo:Performance}(P) \wedge \text{event:factor}(P, \text{:drums}) \wedge \text{mo:recorded\_as}(P, S)) \otimes \text{:average\_tempo}(S, T) \tag{5.6}$$

**Example 3.** To illustrate post-conditions, we express that after inserting an average tempo extracted from a signal in the database, we must check whether it is in a particular range. If the average tempo is not within that range, the database stays in its initial state.

$$\text{:average\_tempo}(\text{:signal}, T) \otimes (T < 130 \ \wedge \ T > 110) \tag{5.7}$$

**Example 4.** We can also use the $\neg$ operator to express further constraints. For example, we can express that inserting an extracted tempo within the database is never allowed, when this tempo is not in a given range.

$$\neg(\text{:average\_tempo}(S, T) \otimes (T > 130 \vee T < 110)) \tag{5.8}$$

[6] An onset detection function provides a basis for detecting the start time of musically relevant events — onsets. For example, such a function includes the spectral difference function, which consists of the magnitude difference between consecutive elements of the STFT transform given in § $A$.2, and its complex counterpart [BDDS04], which also includes information about the phase difference.

We could also express that a transaction must not follow immediately another transaction, that a transaction must follow another transaction, that two transactions must not be executed concurrently.

#### 5.2.2.4 Rules

Rules in $\mathcal{CTR}$ correspond to formulæ of the form $\phi \subset \psi$, where $\phi$ is an atomic formula and $\psi$ is any $\mathcal{CTR}$ formula. As in FOL, $\phi \subset \psi$ is equivalent to $\phi \vee \neg\psi$. The procedural interpretation of such a rule is intuitively "an execution of $\psi$ is also an execution of $\phi$".

We note that if $\psi_1, \ldots, \psi_k$ are non-updating queries, then the rule $\phi \subset \psi_1 \otimes \psi_2 \otimes \ldots \otimes \psi_k$ reduces to $\phi \subset \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_k$. Similarly, $\phi \subset \psi_1 \mid \psi_2 \mid \ldots \mid \psi_k$ reduces to $\phi \subset \psi_1 \wedge \psi_2 \wedge \ldots \wedge \psi_k$.

### 5.2.3 N3-Tr - Interlinked Workflow Descriptions

We define here our representation framework for decentralised workflow descriptions, allowing us to easily combine structured web data and content-based analysis tools. In order to achieve that, we first define in § 5.2.3.1 and § 5.2.3.2 our data and transition oracles. This defines our N3-Tr logic. We then express in § 5.2.3.3 this logic in the N3 syntax described in § 2.2.4.5. We refer to the corresponding knowledge representation framework as N3-Tr. We define N3-Tr *goals* and *rules* in § 5.2.3.4.

#### 5.2.3.1 N3-Tr data and transition oracles

We here create our data and transition oracles, as defined in § 5.2.2.1.

The data oracle $\mathcal{O}^d$ is a mapping from our database states to first-order formulæ. We define a state $\mathbf{D}$ as being a set of RDF triples as defined in § 2.2.4.2. Our state therefore corresponds to the state of an RDF database, or triple store. Our data oracle is then defined as follows.

$$\mathcal{O}^d(\mathbf{D}) = \mathbf{D} \tag{5.9}$$

The transition oracle $\mathcal{O}^t$ is a mapping from pairs of states to the corresponding elementary updates. Our oracle defines a predicate ins, corresponding to the insertion of a ground tuple within the database.

$$\mathsf{ins}(s, p, o) \in \mathcal{O}^t(\mathbf{D}_1, \mathbf{D}_2) \equiv \mathbf{D}_2 = \mathbf{D}_1 \cup \{(s, p, o)\} \tag{5.10}$$

**Example 5.** We consider the following executional entailment, making a database change from a state $\mathbf{D}$ to a state holding two more ground triples.

$$\mathbf{P}, \mathbf{D}, \mathbf{D} + \{(a, b, c)\}, \mathbf{D} + \{(a, b, c), (c, d, e)\} \models \mathsf{ins}(a, b, c) \otimes \mathsf{ins}(c, d, e) \tag{5.11}$$

The truth of a N3-Tr formula is therefore determined on a finite sequence of growing RDF graphs.

#### 5.2.3.2 Built-in and tabled predicates

Moreover, we define a mechanism for adding new elements to our data oracle $\mathcal{O}^d$. We mentioned built-in RDF predicates in § 5.2.1, which can trigger a computation. For example, the spectrogram predicate can trigger a spectrogram computation when queried in the right mode. We extend the definition (5.9) of our data oracle to encompass such built-in predicates as follows.

$$\mathcal{O}^d(\mathbf{D}) = \mathbf{D} \cup \mathbf{D}_b, \text{ where } \mathbf{D}_b = \{\theta | \quad \mathbf{B}, \mathbf{D} \models^b \theta, \ \theta \notin \mathbf{D}\} \tag{5.12}$$

$\mathbf{D}$ corresponds to the database state defined in § 5.2.3.1 — the state of a triple store. Inserts using the ins predicate in the transition oracle operate on $\mathbf{D}$. $\mathbf{D}_b$ corresponds to the set of ground formulæ that can be derived using the built-in predicates available in $\mathbf{B}$ (this derivation is captured by the $\models^b$ operator), and that are not in $\mathbf{D}$. For example, if $\mathbf{B} = \{\mathsf{spectrogram}\}$, $\mathcal{O}^d(\mathbf{D})$ would be the union of $\mathbf{D}$ and all the ground formulæ that can be derived using the spectrogram built-in.

A built-in predicate in $\mathbf{B}$ does not trigger a state change in itself. However, predicates can be *tabled*. In that case, they may trigger state changes when evaluated: we update the database with corresponding evaluations — we *cache* them. We therefore define a new table predicate as follows.

$$\mathsf{table}(S, P, O) \subset (S, P, O) \otimes \mathsf{ins}(S, P, O) \tag{5.13}$$

When applied to a built-in predicate, table can be seen as the transfer of a ground formula from $\mathbf{D}_b$ to $\mathbf{D}$. It stores the results derived using a built-in predicate (hence available in $\mathbf{D}_b$) in the database. Then, the corresponding results are available in $\mathbf{D}$ but no longer in $\mathbf{D}_b$. Subsequent evaluations do not trigger any state change, as the ground formula is already in the current database state $\mathbf{D}$.

The number of elements to insert within the database depends on the determinism of $P$. The rule in (5.13) corresponds to the case where $P$ is deterministic i.e. $P$ is a function. In the case of a non-deterministic predicate, we insert multiple elements in the database, corresponding to the different possible evaluations[7].

We consider the following determinism categories, inspired by the Mercury language [Bec05], and associated to a number of possible sets of outputs given a valid set of inputs. This number corresponds to the number of elements we have to insert in the database when evaluating a predicate in this determinism category.

- *Deterministic* – one solution ;
- *Semi-deterministic* – no solutions or one solution ;
- *Multi-solution* – more than one solution ;
- *Non-deterministic* – zero or more solutions ;

**Example 6.** The spectrogram predicate is deterministic. For one set of input and parameters, it has exactly one output. Evaluating $\mathsf{table}(\mathsf{input}, \mathsf{spectrogram}, S)$ in an empty state will lead us to the state $\{(\mathsf{input}, \mathsf{spectrogram}, \mathsf{output})\}$ and associate the variable $S$ with the spectrogram output. Evaluating $\mathsf{table}(\mathsf{input}, \mathsf{spectrogram}, S)$ in this state will associate $S$ with the previously computed spectrogram output and not trigger any state change.

[7]This is a different behaviour from the non-deterministic transactions described in [BK94]. We here first compute a sequence of database inserts corresponding to each possible evaluation before executing it.

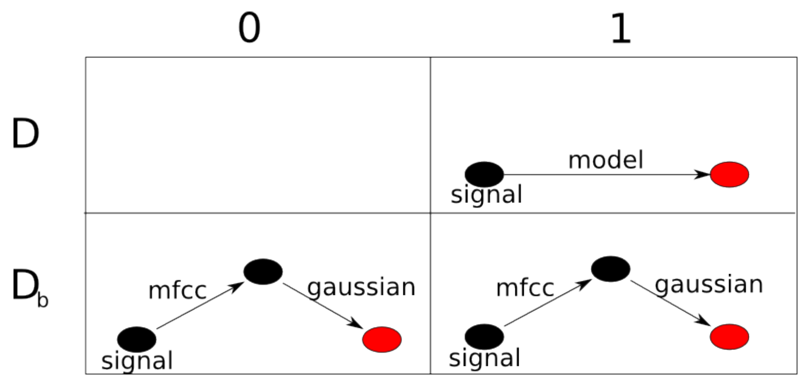


Figure 5.1: Sequence of states over which $\mathsf{table}(\mathsf{signal}, \mathsf{model}, T)$ is true. Our database gets from an empty state to a state holding a model relationship.

**Example 7.** We now illustrate the tabling of non-built-in predicates. We consider two built-in predicates mfcc and gaussian. The first predicate associates an audio signal with Mel-Frequency Cepstrum Coefficients (MFCCs). The second predicate fits such coefficients in a Gaussian model, by deriving their mean and covariance. The computation of MFCCs and of a Gaussian model is further described in § $A.4$. We consider these two predicates as being deterministic. We consider the following transaction.

$$(S, \mathsf{model}, T) \subset (S, \mathsf{mfcc}, M) \otimes (M, \mathsf{gaussian}, T) \tag{5.14}$$

The following formula is true over the sequence of states depicted in Figure 5.1, with the variable $T$ associated to the result of the Gaussian model fitting.

$$\mathsf{table}(\mathsf{signal}, \mathsf{model}, T) \tag{5.15}$$

**Example 8.** We now illustrate non-deterministic transactions. We consider two built-in predicates mfcc and chromagram, respectively associating an audio signal with MFCCs and its chromagram, as described in § $A.4.1$ and § $A.3$.

The following formula has two possible truth values over the sequences of states depicted in Figure 5.2, with the variable $P$ and $O$ respectively associated with mfcc and the outputted MFCCs, or associated with chromagram and the outputted chromagram.

$$\mathsf{table}(\mathsf{signal}, P, O) \tag{5.16}$$

#### 5.2.3.3 N3-Tr logic in the N3 syntax

We defined our N3-Tr logic in § 5.2.3.1 and § 5.2.3.2. We now express this logic within the N3 syntax described in § 2.2.4.5, in order to be able to publish and interlink N3-Tr formulæ specifying music analysis workflows on the Web alongside other structured data in RDF or N3. We define how we express the different N3-Tr constructs in the N3 syntax.

A N3 statement is translated to a N3-Tr formula as follows.

**Example 9.** The N3 statement

```
?p event:factor :drums .
```

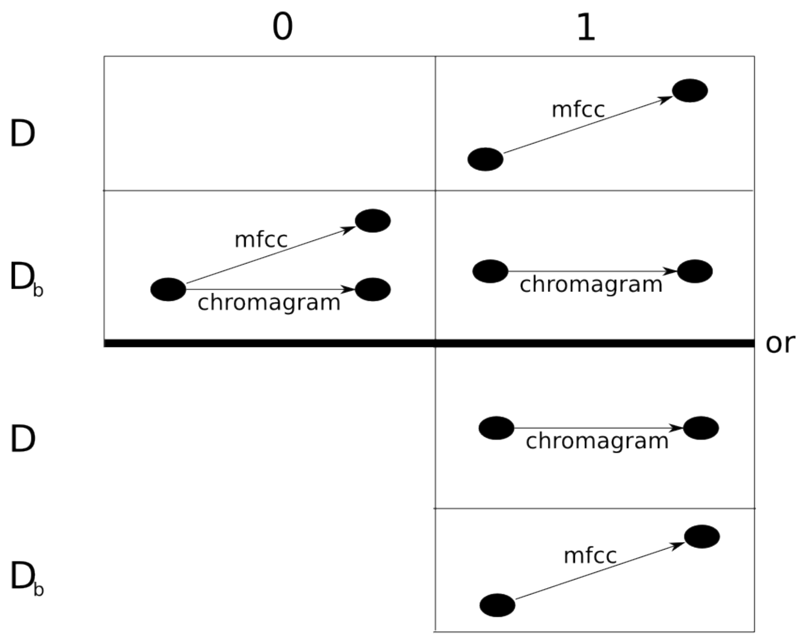


Figure 5.2: Sequences of states over which $\mathsf{table}(\mathsf{signal}, P, O)$ is true. Our database gets from an empty state to a state holding either a mfcc relationship or a chromagram relationship.

corresponds to the following N3-Tr formula:

$$(P, \mathsf{event{:}factor}, \mathsf{{:}drums})$$

We define a web identifier `ctr:TabledPredicate`, capturing the fact that a RDF property is tabled. If a RDF property is tabled, we interpret a corresponding RDF predicate as a table transaction, as defined in § 5.2.3.2.

**Example 10.** If

```
sig:spectrogram a ctr:TabledPredicate .
```

then the following N3 statement

```
:sig sig:spectrogram ?s .
```

corresponds to

$$\mathsf{table}(\mathsf{{:}sig}, \mathsf{sig{:}spectrogram}, S)$$

The determinism categories mentioned in § 5.2.3.2 are captured by the following web identifiers.

- Deterministic – `owl:FunctionalProperty`;
- Semi-deterministic – `ctr:SemiDeterministic`
- Multi-solution – `ctr:MultiSolution`;
- Non-deterministic – `ctr:NonDeterministic`.

**Example 11.** We specify that `sig:spectrogram` is deterministic as follows.

```
sig:spectrogram a owl:FunctionalProperty .
```

A N3 graph literal corresponds to the serial conjunction of the different statements constituting it. The order of the statements in a graph is therefore important. However, as noted in § 5.2.2.4, if none of the statements involve a state change, as in N3Logic [BLCK+08], the serial conjunction reduces to the classical conjunction: the order becomes irrelevant. This ensures that FOL rules (e.g. the rules describing our Music Ontology framework in chapter 3) expressed in N3 are compatible with our framework.

**Example 12.** The N3 graph literal

```
{?a mo:encodes ?s.  ?s sig:spectrogram ?p}
```

corresponds to[8]

$$(A, \mathsf{mo{:}encodes}, S) \otimes \mathsf{table}(S, \mathsf{sig{:}spectrogram}, P)$$

We define three other web identifiers for $\mathcal{CTR}$ constructs: `ctr:cc`, `ctr:or` and `ctr:O`, respectively corresponding to the concurrent conjunction |, the classical disjunction $\vee$ and the isolation operator $\odot$.

**Example 13.** The N3 statement

```
{:signal1 sig:spectrogram ?s1} ctr:cc {:signal2 sig:spectrogram ?s2} .
```

corresponds to

$$\mathsf{table}(\mathsf{:signal1}, \mathsf{sig{:}spectrogram}, S_1) \mid \mathsf{table}(\mathsf{:signal2}, \mathsf{sig{:}spectrogram}, S_2)$$

**Example 14.** The N3 statement

```
{?p event:factor :drums} ctr:or {?p event:factor :congas} .
```

corresponds to

$$(P, \mathsf{event{:}factor}, \mathsf{:drums}) \vee (P, \mathsf{event{:}factor}, \mathsf{:congas})$$

**Example 15.** The N3 statement

```
{:signal :onset_detection_function ?f. ?f :beat_indices ?i} a ctr:O .
```

corresponds to

$$\odot((\mathsf{:signal}, \mathsf{:onset\_detection\_function}, F) \otimes (F, \mathsf{:beat\_indices}, I))$$

We also reuse the web identifier `log:implies`, and the corresponding N3 keywords `=>` and `<=` mentioned in § 2.2.4.5.

**Example 16.** The N3 statement

```
{ ?a :af_spectrogram ?p } <= { ?a mo:encodes ?s. ?s sig:spectrogram ?p } .
```

corresponds to

$$(A, \mathsf{:af\_spectrogram}, P) \subset (A, \mathsf{mo{:}encodes}, S) \otimes \mathsf{table}(S, \mathsf{sig{:}spectrogram}, P)$$

[8]In the following examples, we consider that `sig:spectrogram` is a tabled predicate.

For the $\neg$ operator, we reuse the web identifier `log:Falsehood`.

**Example 17.** The N3 statement

```
{?s :average_tempo ?t. ?t math:lessThan 130; math:greaterThan 110}
  a log:Falsehood .
```

corresponds to

$$\neg(\textsf{:average\_tempo}(S,T) \otimes (T > 130 \vee T < 110))$$

We can now express the N3-Tr logic defined in § 5.2.3.1 and § 5.2.3.2 in the N3 syntax.

#### 5.2.3.4 Graphs, goals and rules

We now define *atomic graphs*, N3-Tr *goals* and N3-Tr *rules*.

**Definition 10.** An atomic graph is a N3 graph literal not holding any graph literals. For example, {`a b c.  d e f`} is an atomic graph whereas {`a b` {`c d e`}} is not.

**Definition 11.** A N3-Tr goal is defined recursively as follows:

- Any atomic graph is a N3-Tr goal;
- {$\mathcal{G}_1$ `ctr:or` $\mathcal{G}_2$} and {$\mathcal{G}_1$ `ctr:cc` $\mathcal{G}_2$} are N3-Tr goals if $\mathcal{G}_1$ and $\mathcal{G}_2$ are N3-Tr goals;
- {$\mathcal{G}$ `a ctr:O`} is a N3-Tr goal if $\mathcal{G}$ is a N3-Tr goal.

**Definition 12.** A N3-Tr rule is a N3-Tr formula of the form $\mathcal{B}$ `log:implies` $\mathcal{H}$. We call $\mathcal{B}$ the *body* of the N3-Tr rule, and $\mathcal{H}$ the *head* of the N3-Tr rule. The body is a N3-Tr goal. The head is an atomic graph holding just one triple[9].

Our N3-Tr rules are *concurrent Horn rules*, and the N3-Tr logic is within the concurrent Horn fragment of $\mathcal{CTR}$. We can therefore use the sound and complete proof theory described in [BK96], which computes new database states as it answers the query.

#### 5.2.3.5 Discussion – what predicates should be tabled?

Our N3-Tr framework encompasses rules that do not involve any state change in the case where none of the predicates within it is tabled. This ensures compatibility with previous rule-based approaches, such as those mentioned in § 5.1.4. However, to get full benefit from the underlying logic, it is important to consider as tabled every predicate that may require a significant amount of computation. Of course, predicates such as `list:in`, associating a list with an element of that list, does not need to be tabled. We can easily re-derive their truth values when needed. But predicates corresponding to a statistical modelling, for example, benefit from being tabled. Another situation where we need predicates to be tabled is when it is difficult to retrieve the inputs from the sole output. We need to keep track of the inputs and algorithm used for the reasons stated in § 5.2.1.

When designing an ontology of such predicates, we need to explicitly identify those which can be tabled.

[9]We will detail in § 7.1.6 how we can still deal with multiple triples and existentially quantified variables in the head of a N3-Tr rule.

### 5.2.4 Examples

We now study three examples of N3-Tr formulæ. Two examples deal with music data, and one example deals with image data to illustrate that this framework is not specific to music processing workflows.

#### 5.2.4.1 Danceable track

Returning to our earlier example (a “danceable” track is a track with an average tempo of around 120 beats per minute, a high loudness, and which is in the playlist of a club disc jockey), we could express it in N3-Tr as follows.

```
:average_tempo a ctr:TabledPredicate.

{?track a :DanceableTrack} <=
  {
    ?playlist a mo:Playlist; mo:track ?track.
    ?playlist foaf:maker ?dj.
    ?dj a mo:DiscJockey.
    ?track mo:available_as ?af.
    ?af mo:encodes ?signal.
    ?signal :average_tempo ?tempo.
    ?tempo math:lessThan 130; math:greaterThan 110.
  }.
```

To categorise a track as “danceable”, we first check if it is included in a disc jockey’s playlist. Then, we look for an audio file corresponding to that track[10]. We then decode this audio file to get to the audio signal. We extract the average tempo from it using the `:average_tempo` tabled predicate. Finally, we make sure this average tempo is within the $110 - 130$ range.

Now, how can we evaluate the `:average_tempo` predicate? It can be part of the available built-in predicates, just as the `sig:spectrogram` predicate mentioned in § 5.2.1. Or the representation of the `:average_tempo` web identifier may hold another N3-Tr workflow (or several), detailing how to derive it.

In order to make the “danceable” workflow accessible, we consider making it part of the structured representation of the `:DanceableTrack` web identifier. Now, a user agent implementing a N3-Tr interpreter, when asked to prove that a track is danceable in that sense, can access this workflow and apply it.

#### 5.2.4.2 Person’s depictions

N3-Tr can be easily applied to other types of media. For example, we consider the following example:

A person $X$ appears in picture $P_2$ if person $X$ is in picture $P_1$ and $P_1$ is similar to $P_2$.

This is expressed in N3-Tr as follows.

[10]We here use the distinction between manifestations and items described in § 3.3.1.2.

```
img:sim a ctr:TabledPredicate.

{?x foaf:depiction ?p2} <=
  {
    ?x a foaf:Person; foaf:depiction ?p1.
    ?p1 img:sim ?p2
  }.
```

This N3 rule specifies that, in order to prove that $?x$ is in picture $?p2$, we have to know that $?x$ is a person (according to the FOAF definition of a person, see § 3.3.1.1) and that he has an image depiction $?p1$. Then, this image $?p1$ has to be similar to another image $?p2$. This last step can be evaluated through the use of a built-in predicate, or `img:sim` may hold in its representation another workflow specifying what this predicate actually captures — an image similarity adapted for finding co-depictions of a person.

#### 5.2.4.3 Content-based audio similarity

We now specify how to derive content-based similarity statements between audio signals. A N3-Tr document defining a concurrent process for deriving a similarity measure between two signals is as follows.

```
sig:mfcc a ctr:TabledPredicate.
sig:gaussian a ctr:TabledPredicate.
sig:jsdiv a ctr:TabledPredicate.

{(?signa1 ?signal2) sim:div ?div} <=
  {
    { ?signal1 sig:mfcc ?mfcc1. ?mfcc1 sig:gaussian ?model1 }
         ctr:cc
    { ?signal2 sig:mfcc ?mfcc2. ?mfcc2 sig:gaussian ?model2 } .

    (?model1 ?model2) sig:jsdiv ?div
  }.
```

First, we derive concurrently two timbre models for the two signals, by extracting their MFCCs and fitting them into a Gaussian model. We then compute a symmetrised Kullback-Leibler divergence between these two models. This rule is therefore specifying the approach described in § $A.4$. The graph available in the database after deriving a single divergence between two content-based models can be depicted as in Figure 5.3. This workflow can be made available as part of a structured representation of `sim:div`.

Now, this other N3-Tr formula can be published elsewhere on the Web, reusing the output of such a workflow to derive Music Ontology information.

```
{?signal1 mo:similar_to ?signal2} <=
        {
          (?signa1 ?signal2) sim:div ?div.
```

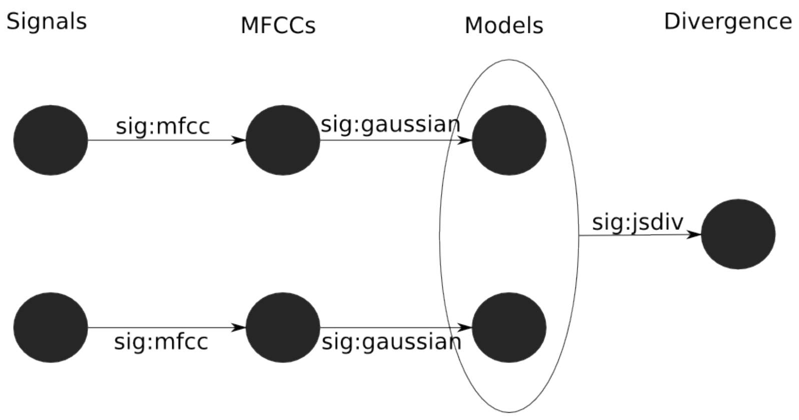


Figure 5.3: Overall graph available in the database for a single divergence between two content-based models, as specified in § 5.2.4.3. Each arrow corresponds to an RDF statement.

```
    ?div math:lessThan 0.3.
  }.
```

The first signal is similar to the second signal if these two timbre models are close enough. Of course, other approaches to derive `mo:similar_to` statement may be specified, such as the approaches described in [Pam06]. Being able to derive similar statements by multiple means illustrates the need for a proof tracking mechanism, as highlighted in § 5.1.5.

### 5.2.5 Discussion – web aspects of N3-Tr

We can now publish N3-Tr rules on the Web, alongside other structured data in RDF or N3. In the previous examples, we saw that the body of such rules can refer to further web identifiers giving access to further information and workflows, in exactly the same way as outlined in § 2.2.3. This results in a sort of 'web of programs' operating transparently on top of the web of data discussed in § 2.2.3, and following the architectural principles outlined in [JW04]. We therefore move towards the "Code on Demand" paradigm [FPV98]. N3-Tr workflows specify the know-how required for processing RDF information and music-related data. Such workflows can be aggregated and interpreted in order to derive new information by combining different analysis tools and structured web data.

### 5.2.6 Provenance tracking

As explained in § 5.1.5, we want to keep track of the different steps leading to a particular result in order, for example, to filter out a derived statement if it relies on an algorithm we do not trust.

Our N3-Tr framework provides a good ground for such provenance tracking. Resources that were created during a workflow can be reverse-engineered, as every tabled computational step leading to a particular result is inserted in the database. We can get back from a newly created resource to the inputs, parameters and algorithm used, as in our spectrogram example in § 5.2.1. The resulting database is fully self-describing. Our N3-Tr logic embeds the tracking of workflow instances, as defined in § 5.1.2. Moreover, by wrapping the different computational steps in a graph literal, we can keep track of further information about them, e.g. the duration of the computation, when it was performed, or the machine that was used.

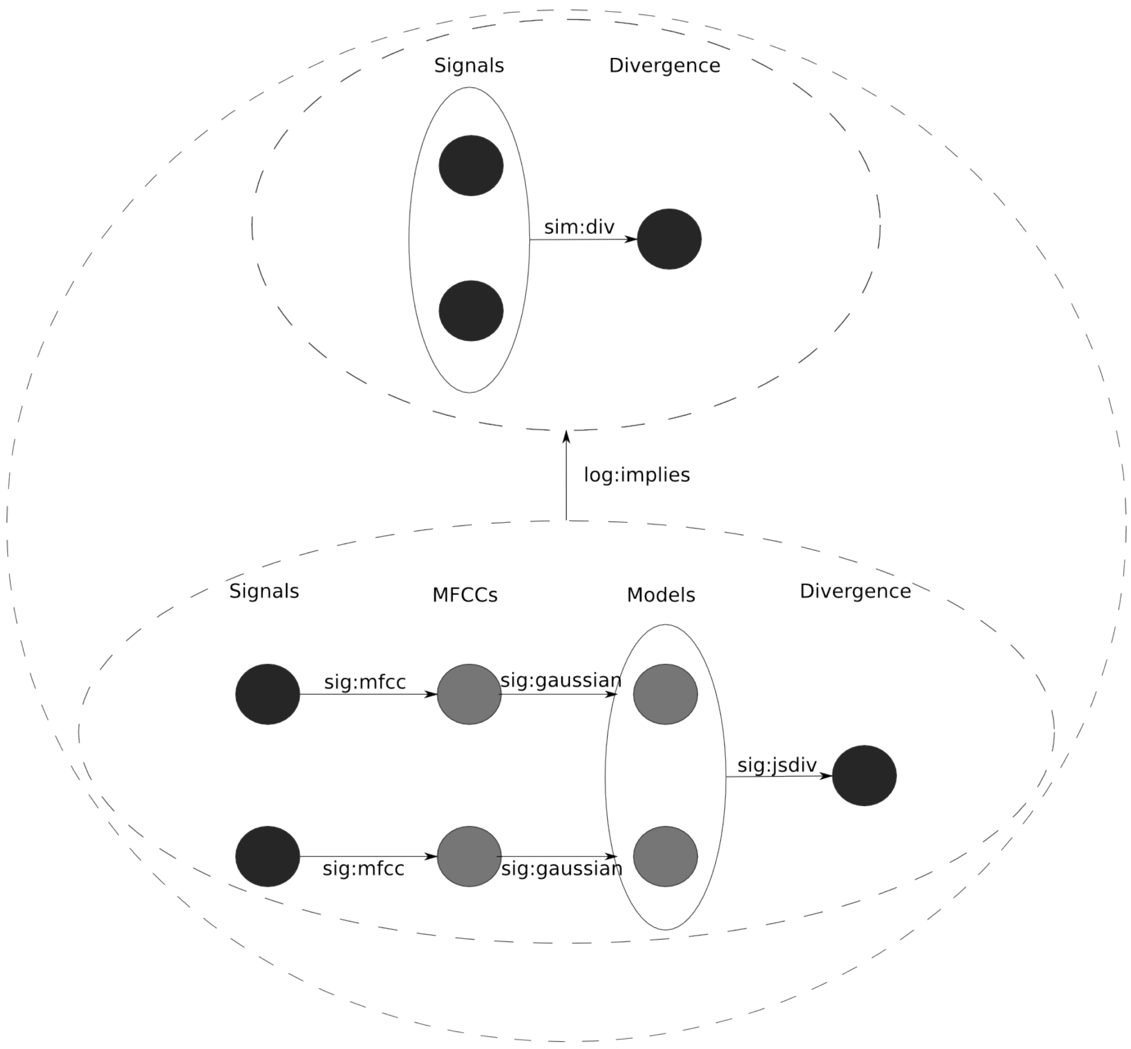


Figure 5.4: Applying the N3-Tr rule in § 5.2.4.3. A dotted line denotes a graph literal.

In our audio similarity example described in § 5.2.4.3, we could get back from a divergence to the algorithm applied (`sig:jsdiv`, identifying a symmetrised and normalised Kullback-Leibler divergence) and to the inputs (two content-based models). We can also get back from these inputs to the process used to derive them. All that defines the meaning of that divergence. The overall graph available in the database when a single divergence is computed can be depicted as in Figure 5.3.

Intuitively, a N3-Tr rule provides a view over such a graph. For the graph depicted in Figure 5.3, when applying the first N3-Tr rule in § 5.2.4.3, we directly link the two signals to the derived divergence through a `sim:div` predicate, as depicted in Figure 5.4.

The provenance tracking capabilities of N3-Tr allows us to apply a wide range of information filtering policies [Biz07, BC08], e.g. "I do not trust statements which depend on that particular type of low-level feature".

Provenance tracking is a really important matter, and the problem is not only to have access to such information, but also to interpret and propagate it. Also, our provenance tracking is specific to a particular kind of derivation. In order to handle other kinds of derivation, we would need more complex provenance tracking systems, such as the system described in [MdS04]. It remains future work to investigate these issues within our N3-Tr framework.

## 5.3 On-demand publishing of derivable information

We now have a framework for publishing and interlinking music analysis workflows expressing how structured data can be combined with analysis tools to produce more information. We now focus on on-demand computation and publishing of information that can be derived by applying such workflows. This will complete our framework by allowing us to reuse derived information in further workflows or in a wide range of applications, from personal collection management to recommendation.

Since automated analysis tasks such as those in § 5.2.4 may require a significant amount of computation, some care must be taken to avoid wasted effort. The naïve approach of precomputing all the derivable information can be dismissed immediately. We therefore focus on providing an on-demand access to such derivable information.

Structured data on the Web can be accessed in two different ways: through a SPARQL query, as mentioned in § 2.2.5, or through retrieving the representation of a web resource from its identifier. These two interfaces can be used to access information which is computed only when requested, perhaps derived by a component interpreting aggregated N3-Tr formulæ. In the music realm, this means that current research algorithms for tempo and rhythm estimation, harmonic analysis, partial transcription, or source separation could be exposed on the Semantic Web. This is of great benefit both to researchers, allowing them to more easily compare or build upon others' algorithms and to the general public by letting them use research algorithms without requiring each researcher to design end-user applications. This provides a possible answer to the concerns expressed by Boll in an image analysis context [Bol07]:

> "Imagine you work on semantic annotation of photos and aim to automatically annotate a photo with names, places, and events. But you aren't the only one in the field, and you would like to benchmark your own work against another group's data set, or you aim to reuse a third group's method. Today, there are almost no good standards and procedures in our community to share and reuse data, methods, models, or algorithms—this is something we should change!"

We describe in this section our two mechanisms for providing an on-demand access to derivable information. These mechanisms can be used by N3-Tr interpreters to provide an on-demand access to the information they are able to derive, or by any other component having the ability to derive new information (algorithms, Web Services, inference engines, etc.). These mechanisms also enable the auto-discovery of derived information by user agents. We describe a SPARQL interface in § 5.3.1 and a web interface in § 5.3.2. We discuss further implications of our N3-Tr framework and our publication mechanisms in § 5.3.3.

### 5.3.1 On-demand access to derived data through SPARQL

We describe here our mechanism for publishing derivable information through SPARQL.

#### 5.3.1.1 N3-Tr and SPARQL

SPARQL provides support for Named Graphs, as defined in § 2.2.4.2. Triple patterns can be queried within a particular context, itself explicitly named by a web identifier. Using a similar mapping as defined in § 5.2.3.3, we can use SPARQL as an alternative syntax for N3-Tr goals[11].

For example, we can rewrite the N3-Tr goal given in § 5.2.4.2 as a SPARQL query.

```
SELECT ?person ?implicit_depiction
WHERE {
   ?person a foaf:Person; foaf:depiction ?depiction.
   ?depiction img:sim ?implicit_depiction.
}
```

We first lookup the depiction of a person, and we then compute similar images.

We can rewrite the concurrent N3-Tr goal given in § 5.2.4.3 as a SPARQL query.

```
SELECT ?div
WHERE {
   GRAPH ?g1 { :signal1 sig:mfcc ?mfcc1. ?mfcc1 sig:gaussian ?model1 }
   GRAPH ?g2 { :signal2 sig:mfcc ?mfcc2. ?mfcc2 sig:gaussian ?model2 }
   ?g1 ctr:cc ?g2 .
   (?model1 ?model2) sig:jsdiv ?div .
}
```

We derive concurrently two timbre models for the two signal `:signal1` and `:signal2`, and we compute a divergence between these two models.

#### 5.3.1.2 Derived information and SPARQL

A SPARQL interface can also be used to directly access the information derived by applying a N3-Tr rule. In that case, the fact that the end-point gets through some steps to generate the information is completely transparent to the user agent.

For example, the following SPARQL query could trigger the workflow defined in § 5.2.4.2.

```
SELECT ?depiction
WHERE
{ <http://moustaki.org/foaf.rdf#moustaki> foaf:depiction ?depiction }
```

The following SPARQL query could trigger the N3-Tr workflow defined in § 5.2.4.3.

```
SELECT ?div
WHERE
{ (:signal1 :signal2) sim:div ?div }
```

The corresponding divergence would be associated to the variable `?div`. As described in § 5.2.6, in the case the N3-Tr workflow in § 5.2.4.3 was used to derive this statement, more information

[11]We could also use SPARQL as an alternative syntax for N3-Tr rules, using `CONSTRUCT` queries of the form `CONSTRUCT` *head* `WHERE` *body*, in a way similar as in [Pol07].

about this divergence can be queried — tabled computations would have populated the underlying database when creating the divergence. For example, we could get back to the workflow leading to the creation of that divergence (content-based models and symmetrised Kullback-Leibler divergence between these). For example, we could use such information to restrict the returned divergence to be based on MFCCs. We can also use the support for Named Graphs in SPARQL to query more information about the N3-Tr derivation step leading to this `sim:div` statement, as depicted in Figure 5.4.

A SPARQL end-point having a particular capability can expose it using the mechanism we describe in Appendix B. Then, a component aggregating such capability descriptions can redirect a user agent towards a SPARQL end-point which would help answer its query, as illustrated in B.3.

### 5.3.2 On-demand access to derived data on the Web

Our second access mechanism is based on the idea that accessing a web resource may trigger some computation, for example a N3-Tr workflow or a particular algorithm. We describe two variants of this access mechanism.

#### 5.3.2.1 Advertisement statements

Our first approach to exposing algorithms or workflows on the Web is immediately compatible with existing user agents, such as the user agents mentioned in § 2.2.5, while still avoiding wasteful computation. We consider publishing *advertisement* statements, providing user agents with a web identifier which, when accessed, will trigger a unique class of computation. The property used in these advertisement statements is the property they advertise. For example, if we know how to derive chromagrams and MFCCs from audio signals, we publish the following advertisement statements when the description of an accessible signal `:signal` is requested.

```
:signal sig:chromagram :chroma_adv.
:signal sig:mfcc :mfcc_adv.
```

Then, when a user agent accesses `:chroma_adv` or `:mfcc_adv`, we execute the corresponding process, and we append the resulting statements to the returned description. For example, if the user agent is looking for a chromagram, it follows the `sig:chromagram` link and accesses `:chroma_adv`, which triggers the chromagram computation. The user agent then gets back the actual chromagram.

When considering a resource identifying an audio signal as defined in § 3.3.2.2, we might create multiple advertisement statements corresponding to lots of different potentially derivable data. Our approach ensures that computation effort is not wasted. Only computations that suit a Semantic Web user agent request are triggered.

#### 5.3.2.2 Dynamic documents

Another approach which has been suggested[12] is that within the representation of a resource (a signal, for example), we use a specialised vocabulary to link to an RDF document. This vocabulary gives the user agent some indication of what it would find out by retrieving this

[12]in the "Limitations on browseable data" section of [BL06a], although in a different context

document. The user agent may then make an informed decision as to whether to retrieve the extra information or not.

For example, a specialised predicate `mo:onset_document` subsuming `rdfs:seeAlso` could indicate "more information about the onsets detected in this signal can be found in that document". If the user agent is a beat tracker operating on onsets as an input, it would retrieve this extra document, whereas if the user agent is an editorial information crawler it would not.

Compared with the advertisement statement approach presented in § 5.3.2.1, the main disadvantage is that it requires user agents to understand the semantics of such links in order to fully browse and query all available information.

The main advantage is that the structure is explicit and left to the discretion of the user agent, which might interpret it in unexpected and useful ways. Another advantage is that this mechanism is much easier to set up. Such RDF documents can be constructed using SPARQL queries, as in § 5.3.1.

Information about the derivation process can be included in the approaches described in § 5.3.2.1 and § 5.3.2.2. We consider an RDF graph as being the set of statements gathered when accessing a web identifier. Then, we can attach derivation information to that web identifier. For example, the following graph corresponds to the information derived by the rule in § 5.2.4.3, as depicted in Figure 5.4.

```
:premises log:implies <>.
(:signal1 :signal2) sim:div :divergence.
```

As most Semantic Web user agents keep track of the origin of the RDF statements they gather using the Named Graphs mechanism described in § 2.2.4.2, they are able to relate such derivation information to the derived statements.

#### 5.3.2.3 Example: Content-based audio similarity

We now take the example described in § 5.2.4.3, and publish the corresponding derivable information using the mechanism described in § 5.3.2.1.

We consider a N3-Tr interpreter holding the first rule specified in § 5.2.4.3. This interpreter is able to derive a content-based divergence measure between two audio signals. This first interpreter publishes information in the namespace `i1`. We consider a second N3-Tr interpreter holding the second N3-Tr rule specified in § 5.2.4.3. This interpreter is able to threshold a similarity measure to derive `mo:similar_to` statements. The second interpreter publishes information in the namespace `i2`.

The first interpreter has access to a repository of audio signals. For a pair of them, `i1:sig1` and `i1:sig2` in our example, it publishes an advertisement statement, linking to a new web identifier `i1:adv`. This identifier corresponds to the result of the content-based audio similarity process. The following statement is made available on the Web.

```
(i1:sig1 i1:sig2) sim:div i1:adv.
```

For all pairs of signals, a similar statement is published. Now, we consider that our second interpreter aggregated such statements. The second interpreter can build on top of such information, using the second rule in § 5.2.4.3. It then publishes the following advertisement statement, corresponding to a signal similar to `i1:sig1` obtained by thresholding the similarity measure advertised by the first interpreter.

```
i1:sig1 mo:similar_to i2:adv.
```

For all signals, a similar statement is published. The advertisement statements published by the first and the second interpreter serve to define the computation network.

Now, a user agent wanting to know more about signals similar to `i1:sig1` accesses `i2:adv`. This triggers the thresholding process in the second interpreter. This process needs the actual value of the content-based divergences, so it accesses the divergence resources, including `i1:adv`. Accessing these resources triggers the content-based divergence process in the first interpreter.

The first interpreter delivers the content-based divergences. The second interpreter delivers the similarity statements. For example, accessing `i2:adv` gives the following.

```
sw:sig1 mo:similar_to sw2:adv.
sw:sig1 mo:similar_to sw1:sig37.
sw:sig1 mo:similar_to sw1:sig41.
sw:sig1 mo:similar_to sw1:sig132.
sw:sig1 mo:similar_to sw1:sig211.
sw2:adv owl:sameAs sw1:sig31.
```

In order to make it totally transparent for the user agent, we also state that `i2:adv` is the same as one of the matching outputs. The resource accessed by the user agent is one of the similar signals.

An implementation of this example is described in § 7.3.2.

### 5.3.3 Discussion – distributed music processing workflows

In this last example, we see what our N3-Tr framework along with our publication mechanisms leads to. A new algorithm and the structured nature of its output is shared by publishing a N3-Tr document, stating how to stack different operations to derive further information. Information is derived by an N3-Tr interpreter only when needed, either by a user agent directly exploiting it e.g. for visualisation purposes, or by another analysis workflow building upon it.

If computation delay is not a significant issue, we can go further in the distribution of the content-based similarity. Each step (MFCCs computation, modelling, divergence and thresholding) can be spread across different hosts. Moreover, if any of the hosts have aggregated different adverts for the same information they are free to substitute those corresponding information providers, and so the computation network is flexible and dynamic.

In our particular setup, only the first interpreter needs access to the audio content. The other hosts only deal with higher-level representations of the underlying signal. A topic of ongoing discussion in the digital music research community is how to perform analysis across very large music data sets when any one laboratory has access to only a subset of the audio data (due to copyright considerations). Some argue that higher-level representations (e.g. rhythm and timbre information which cannot be used to reconstruct the audio) derived from the copyright signal are themselves not covered by the copyright, in which case the system described above exhibits a clear advantage, allowing any number of systems to derive additional information about music tracks without violating copyright by accessing the audio itself. Unfortunately, the extent to which these derived representations really are copyright-free is currently unclear. These legal issues are outside the scope of this thesis.

## 5.4 Discussion and related work

In this section, we discuss our framework and its relationships to other work.

### 5.4.1 Web Services

As highlighted in § 5.1.3, providing Web Service access to processing components or information sources is a first step towards a distributed framework to perform multimedia analysis tasks.

However, Web Services require significant effort to combine functionality from several providers. They give back a set of results without any details on what these results are i.e. how they were derived and what has been processed. We therefore do not have a way to automatically interpret these results. Some binding code has to be written, adapting to the peculiarities of the services one is trying to combine.

This problem is avoided in the information-centric approach we present here, as an N3-Tr rule explicitly specifies what it needs to derive new information. The user agent interpreting this rule is free to meet those requirements with suitable information from any provider. A N3-Tr rule also explicitly specifies the information it is able to derive, therefore allowing it to be automatically reused, by other user agents or in other N3-Tr rules.

Of course, specifying the information requirements of an algorithm (e.g. working in the time of frequency domain, with a particular kind of onset detection function) needs a suitably expressive ontology, as highlighted in [PR07]. A number of terms within our Music Ontology framework allows us to make such descriptions, as reviewed in § 3.3.4.5.

Also, Web Services can be captured within our N3-Tr framework. A service can be modelled as a link — a RDF predicate linking its inputs and its outputs. For example, we can wrap the Echonest Analyze API [ech] in a predicate linking an audio file to the extracted structure, rhythm, harmony, keys, and timbre. Then, a N3-Tr rule can give semantic value to this information by mapping it to the corresponding Music Ontology terms. Any direct translation of Web Services results to RDF also allows Web Services to be reused in our N3-Tr framework. For example, we provide a GRDDL transform (as described in § 2.2.4.4) for the Echonest Analyze API[13], allowing user agents to map the results of this Web Service to Music Ontology RDF.

### 5.4.2 Semantic Web services

Semantic Web services provide a way of making a formal description of Web Services. The two main frameworks for these descriptions, built around similar concepts, are OWL-S and WSMO. Services are considered as primary objects, with associated inputs, outputs, parameters, preconditions and effects. On top of these descriptions, we can build automated agents combining and mediating the corresponding services to achieve a particular goal [MSZ01], such as the "semantic broker" described in [DCG+08]. The same comparison as in [BK95] between transaction logic and the situation calculus [MH69] holds to compare our work with Semantic Web services, as our work is based on the former and Semantic Web services are based on the latter [MSZ01].

> "Whereas the situation calculus focuses on specifying the effects of primitive actions, [transaction logic] focuses on combining such actions into complex ones and executing them."

[13]see `http://dbtune.org/echonest/`

Recent work used the formalism of $\mathcal{CTR}$ for tackling the problems of choreography, contracting and service enactment [RK07, RKF08] for Semantic Web services. However, $\mathcal{CTR}$ is used there as a background unifying logic for capturing a number of aspects that were previously requiring separate mechanisms.

In N3-Tr, the $\mathcal{CTR}$ constructs are explicit. Combining two services is as simple as combining sources of information, which may trigger side-effects e.g. computing a spectrogram, a content-based similarity distance or decoding audio items available on the Web. Tasks and information sources are treated uniformly. This allows us to merge factual statements ('this person is in this picture') and tasks results ('this picture is similar to that picture') in a simple way.

In the following, we give an example illustrating how such an information-centric approach compares to a Semantic Web Service approach for the depiction example in § 5.2.4.2. This example could be modelled as follows in a Semantic Web services context.

**Person-depiction-goal**

`"This goal returns depictions of persons"`

**Input role**

`has-person person "URI"`

**Output role**

`has-depiction image "URI"`

**Post Condition**

`the role value of has-depiction is the depiction of the role value of has-person`

**Similar-images-goal**

`"This goal returns images similar to an image"`

**Input role**

`has-image image "URI"`

**Output role**

`has-similar-image image "URI"`

**Post Condition**

`the role value of has-similar-image is similar to the role value of has-image`

In our framework, this corresponds to the two following N3-Tr goals.

```
?person foaf:depiction ?image.
?image img:sim ?similar_image.
```

These two predicates can be combined in an N3-Tr rule to derive a higher-level predicate as in § 5.2.4.2. A planner working on top of the two Semantic Web services above would be able to solve the goal "find me images similar to the depictions of that person", but would be unable to derive such a high-level predicate. We would have to get through another step: defining a workflow involving those two Semantic Web services, using for example the OWL-WS ontology [BCG$^+$05] or a Semantic Web services adaptation of BPEL4WS [MM03].

The *discovery* mechanism in Semantic Web services [MSZ01] is related to the publication part of our framework, described in § 5.3. We described for each of our publication mechanisms how automated discovery can be achieved. The SPARQL publishing approach in § 5.3.1 can use the mechanism we describe in Appendix B. The approaches described in § 5.3.2 embed such a discovery capability, as illustrated in § 5.3.2.3.

In order to cover more of the Semantic Web Services use-cases, our N3-Tr framework would also need another predicate *rdf.del* in its transition oracle, corresponding to the deletion of a particular RDF triple. We would then be able to model deletions and updates (e.g. "buying this airplane ticket removes £500 from a bank account"). It remains future work to study whether such an extension of our N3-Tr framework would be expressive enough to handle Semantic Web services use-cases.

### 5.4.3 Scientific workflow systems

Several editors and sharing services exist for scientific workflows involving multiple Web Services and information sources. For example, the Kepler system [LAB+06] for scientific workflow management allows a scientist to set up an experiment involving remote data sources and computational resources in a graphical environment.

The MyExperiment service [dRGS07] allows scientists to publish and share workflows written in the Scufl language [OGA+06]. A Scufl workflow specifies the orchestration of Web Services. These workflows are concrete, as defined in § 5.1.3. They are bound to specific Web Services. N3-Tr workflows are abstract. A user agent interpreting a N3-Tr workflow can use suitable information from any provider at any step.

The bioinformatics Taverna workbench [OGA+06] interprets Scufl workflows, allowing anyone to reproduce a particular experiment. Taverna also provides the user with a graphical user interface to design, execute and export new workflows orchestrating a set of Web Services. Taverna has a lot of interesting features, including fault tolerance, resilience, monitoring of workflows and graphical editing of workflows. Kepler [LAB+06] is a similar system for scientific workflow management, allowing scientists to set up experiments involving remote data sources and computational resources in a graphical environment. Our implementation of an N3-Tr interpreter, which we describe in chapter 7, focuses on the aggregation, interpretation and publication steps. It does not provide an interface for the editing of N3-Tr rules.

### 5.4.4 Semantic Web rule languages

Several languages other than N3 can be used to describe rules operating on top of RDF information, including SWRL [HPSB+04] and RuleML[14]. It remains future work to investigate if our N3-Tr logic could be expressed in these syntaxes.

An interesting problem is the combination of rule languages and the Description Logics family of languages reviewed in § 2.1.6. We used OWL-DL (corresponding to the $\mathcal{SHOIN}(D)$ description logic) for specifying our Music Ontology framework in chapter 3, and it would be interesting to combine reasoning based on our ontology and N3-Tr reasoning. Similar problems led to Description Logics Programs (DLP [GHVD03]), corresponding to the Description Horn Logic fragment of the $\mathcal{SHOIN}(D)$ description logic. Samuel et al. [SOS+08] describe how DLP can be used as a

[14] `http://www.ruleml.org/`

basis for the integration of Semantic Web rules and Description Logics ontologies. DLP can also be used to integrate Description Logics and N3-Tr. We translate the ontology to DLP axioms, which correspond to non-updating N3-Tr rules. For example, the different axioms specifying our Music Ontology framework in chapter 3 can be re-written as non-updating N3-Tr rules[15]. However, it remains future work to detail such an integration.

## 5.5 Summary

In chapter 3, we defined a framework to publish heterogeneous music-related data on the Web, such as:

- Editorial data (e.g. release, artist, track);
- Music production workflow (e.g. composition, arrangement, performance, recording);
- Event decomposition and temporal annotations (e.g. a performer playing a particular instrument in a performance, chords in a particular work).

In this chapter, we designed a representation framework, N3-Tr, allowing us to express how we can combine different analysis processes and such structured music-related data to derive new information. We then described our publication mechanisms, allowing us to provide an on-demand access to derivable information. The derived information can be directly used by user agents, for example to answer queries such as ‘Give me performances of that Jazz piece where the piano solo includes this musical phrase’, or can be used in another N3-Tr rule, building upon it to enable the derivation of more information.

We are now able to gather music-related information, N3-Tr music analysis workflows and derived information resulting from these workflows in a decentralised and arbitrarily large dataset—the Web. We have the necessary technologies to create autonomous music-related agents, that analyse structured data available on the Web to derive new information, which can then be used for a wide range of purpose e.g. collection management, visualisation, search or recommendation. We will discuss such an autonomous music-related agent in chapter 7 and such applications in chapter 8.

[15]The FOL notation used in chapter 3 makes this mapping straight-forward.

# Part III

# Applications

# Chapter 6

# A web of music-related data

In this chapter, we detail interlinked music-related datasets, published using the Music Ontology framework described in chapter 3. We here focus on “static” data: information is stored somewhere and looked up when requested. The described web of data is, to the extent of our knowledge, the largest available distributed music-related database.

We review in § 6.1 a community project using the above described technologies to publish and interlink large amounts of structured data on the Web. We describe in § 6.2 a set of tools we used to publish and interlink music-related datasets on the Web. We include several of our own tools, namely P2R and UriSpace. We describe in § 6.3 how those tools were practically applied to make a wide range of music datasets available on the Web, within the scope of the community project mentioned in § 6.1. Finally, we quantify some aspects of available structured web data in § 6.5.

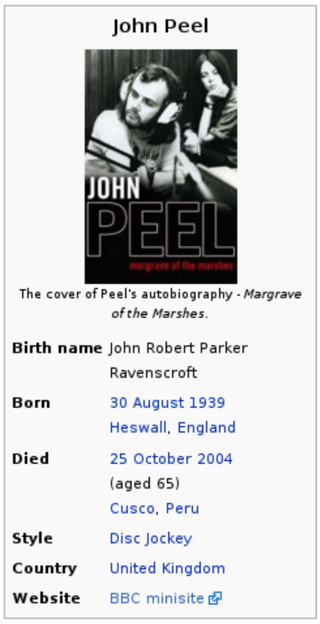
## 6.1 The ‘Linking Open Data on the Semantic Web’ community project

### 6.1.1 Open Data

The *open data* movement aims at making data freely available to everyone. The data sources published cover a wide range of topics: from music (Musicbrainz, Magnatune[1] and Jamendo[2]) to encyclopedic information (Wikipedia[3]) or bibliographic information (Wikibooks[4], DBLP bibliography[5]). We contributed to the ‘Linking Open Data on the Semantic Web’ community project [BHAR07] of the W3C Semantic Web Education and Outreach group[6], which aims at making such data sources available on the Semantic Web, and creating links between them using the technologies described in § 2.2.3.

For example, when we provide a description of an artist in the DBTune project[7], we link the artist resource to a location in Geonames[8] instead of providing a complete geographic description

[1]`http://magnatune.com/`
[2]`http://www.jamendo.com`
[3]`http://wikipedia.org/`
[4]`http://wikibooks.org/`
[5]`http://dblp.uni-trier.de/`
[6]`http://www.w3.org/2001/sw/sweo/`
[7]`http://dbtune.org/`
[8]which provides additional knowledge about this location, e.g. hierarchical relationships with other geographical entities, latitude, longitude. Geonames is available at `http://geonames.org/`.


**John Peel**

The cover of Peel's autobiography - *Margrave of the Marshes*.

| | |
|---|---|
| **Birth name** | John Robert Parker Ravenscroft |
| **Born** | 30 August 1939<br>Heswall, England |
| **Died** | 25 October 2004<br>(aged 65)<br>Cusco, Peru |
| **Style** | Disc Jockey |
| **Country** | United Kingdom |
| **Website** | BBC minisite |

Figure 6.1: Wikipedia *fact box* for the BBC radio 1 disc-jockey John Peel

ourselves. Then an agent crawling the Semantic Web can follow the link from our knowledge base to the Geonames one by following this link.

### 6.1.2 Interlinking heterogeneous data sources

The datasets published in March 2008 by the Linking Open Data community are depicted in Figure 6.2. They cover a wide range of topics. The DBpedia [ABL+07] project extracts structured information from *fact boxes* within the Wikipedia community-edited encyclopedia. Such a fact box is depicted in Figure 6.1. The RKB Explorer dataset [GM07] publishes information about researchers, research projects and publications. The US Census dataset [Tau07] publishes statistical information gathered by the Census Bureau in the United States. The Revyu web site [HM07] allows its users to review and rate any kind of resource, and publishes the corresponding information on the Semantic Web. Overall, over 35 datasets have been published and interlinked within this project and the number is growing all the time.

Creating bridges between previously independent data sources paves the way towards a large machine-processable data web, gathering interlinked Creative Commons licensed content[9], encyclopedic information, domain-specific databases, taxonomies, cultural archives, and so on.

### 6.1.3 Music-related data

The Music Ontology described in chapter 3 supports this process for music-related information. It provides a framework for publishing heterogeneous music-related content in RDF. Moreover, it can be used alongside other ontologies covering other domains, as mentioned in § 3.4.2. For example, we can use the Music Ontology alongside ontologies for reviews, social networking information or geographic information.

[9]See `http://creativecommons.org/`

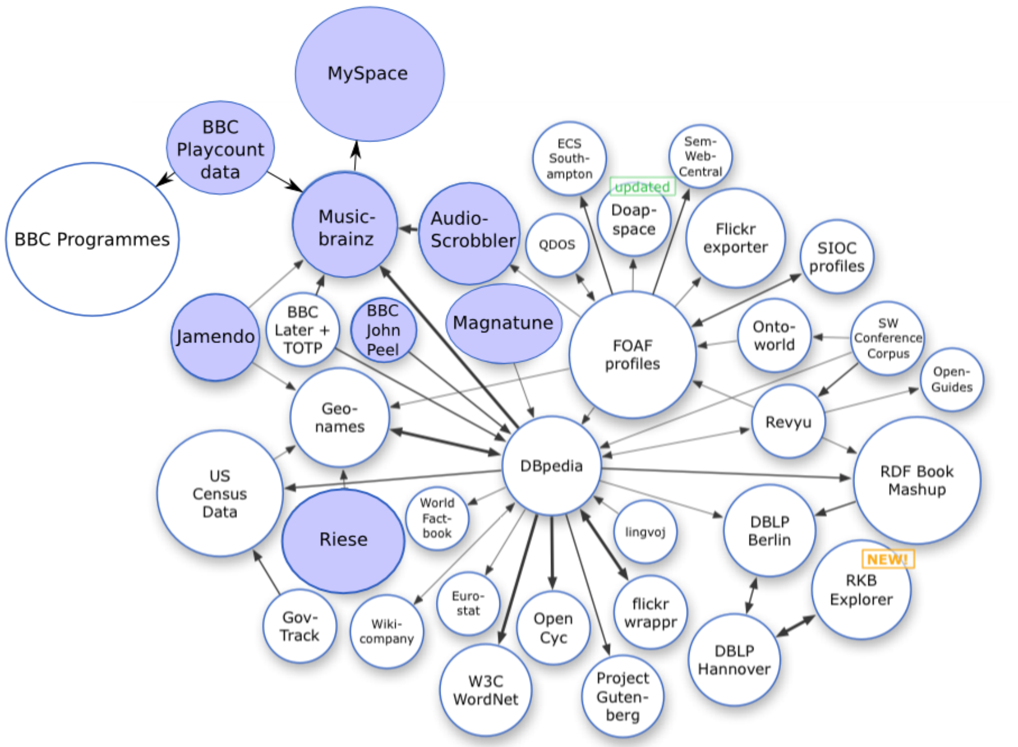


Figure 6.2: Datasets published by the Linking Open Data community, March 2008. Each node corresponds to a particular dataset. Arrows between nodes correspond to links from one dataset to another. Our contribution is highlighted in blue. Diagram by Richard Cyganiak, and available on-line at `http://richard.cyganiak.de/2007/10/lod/`

## 6.2 Tools for publishing datasets on the Web

We describe here a number of tools facilitating the publication of structured data on the Web. In particular, we describe two tools we developed: P2R and UriSpace.

### 6.2.1 D2R Server

D2R Server [BC06] provides a simple way of publishing relational databases on the Semantic Web. A declarative RDF-based mapping language (D2RQ [BS04]) is interpreted by a D2R server, which then provides a SPARQL access and a linked data access. All queries to the SPARQL end-point are translated to relational queries using the D2RQ mapping.

For example, if we have an `artist` table in a relational database, with a primary key `gid` and a column `name`, we can map it to Music Ontology RDF using the following D2RQ RDF code:

```
map:artist a d2rq:ClassMap;
        d2rq:dataStorage map:database;
        d2rq:uriPattern "artist/@@artist.gid@@";
        d2rq:class mo:MusicArtist;
        .
map:artist_name a d2rq:PropertyBridge;
        d2rq:belongsToClassMap map:artist;
        d2rq:property foaf:name;
        d2rq:column "artist.name";
        .
```

Then, a tuple $(1, \mathsf{Weezer})$ in the database would be translated as:

```
<artist/1> a mo:MusicArtist; foaf:name "Weezer".
```

### 6.2.2 P2R

In order to publish datasets in a variety of formats on the Web, we created the Prolog-2-RDF software (P2R, available as a module for SWI-Prolog [WHM08]). P2R translates Prolog predicates to RDF dynamically, when SPARQL queries are issued to the P2R SPARQL end-point. As in D2R, P2R is powered by a declarative mapping from Prolog predicates to a set of RDF triples. However, unlike in D2R, Prolog predicates can themselves wrap a variety of data sources, from relational databases to web services or spreadsheets. This variety of supported data sources is provided through a set of SWI-Prolog modules.

For example, if we have an `artist(Id,Name)` predicate, we can map it to RDF and the Music Ontology using the following P2R Prolog mapping:

```
match:
   artist(Id,Name)
      eq
      [
          rdf(pattern([‘artist/’,Id]),rdf:type,mo:‘MusicArtist’)
      ,   rdf(pattern([‘artist/’,Id]),foaf:name,literal(Name))
      ].
```

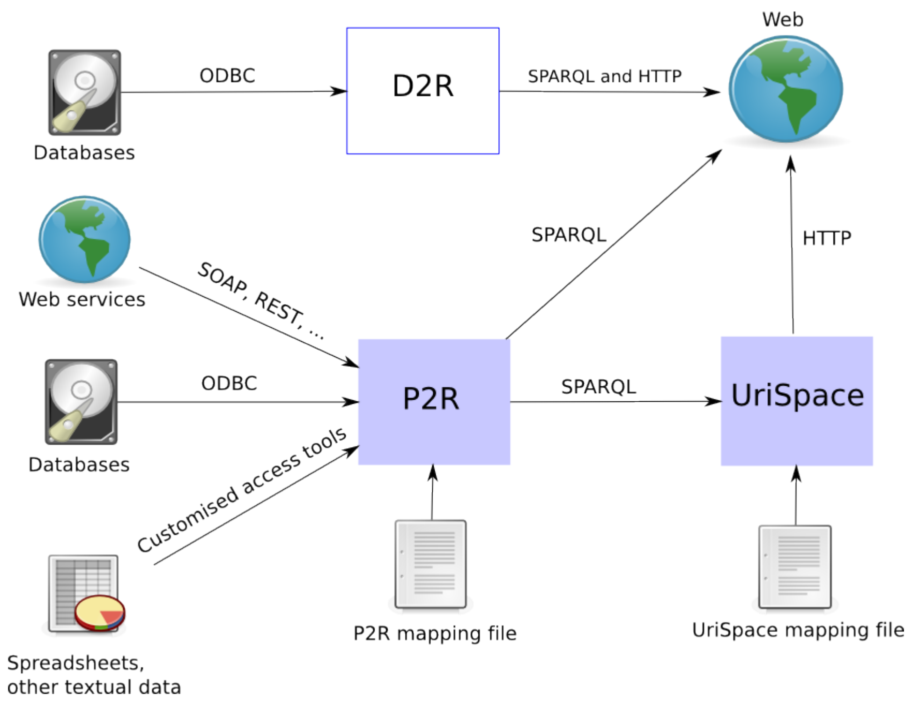


Figure 6.3: D2R, P2R and UriSpace—publishing heterogeneous data as linked data on the Web

Then, if `artist(1,'Weezer')` is true (the evaluation itself can wrap a call to a web service, for example), it would be translated as in § 6.2.1.

### 6.2.3 UriSpace

As P2R provides only a SPARQL end-point to query the resulting RDF data, we need a layer on top of it for the actual publication as structured web data—web identifiers and corresponding structured representations. We designed a small SWI-Prolog module to handle the publication of web data on top of any SPARQL end point: UriSpace. A declarative mapping associates a web identifier pattern for a set of individuals with a SPARQL CONSTRUCT or DESCRIBE query pattern that can generate their structured representations. UriSpace sends back an HTTP 303 redirect towards the results of the corresponding SPARQL query when an individual's identifer is accessed [SC08]. UriSpace also gives the ability to construct web identifiers for informational documents. For example, a web resource could correspond to a list of all available instances of a particular concept.

The overall P2R and UriSpace architecture for the publication of structured web data can be depicted as in Figure 6.3. It compares to D2R in that it provides wrappers for heterogeneous data sources—it is not bound to relational databases.

### 6.2.4 Other publication tools

Several other publication tools exist, such as the OAI2LOD server [HS08], converting digital library data published using the Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) to structured web data. Other publication tools add Semantic Web capabilities to existing platforms, such as Semantic MediaWiki [KVV06] or the various SIOC [BBP$^+$08] exporters[10].

[10] `http://sioc-project.org/exporters`

## 6.3 Publishing real-world datasets

We used the tools described in § 6.2 in a number of real-world publication and interlinking scenarios. The datasets described in this section are available as part of our DBTune service [Rai]. All the code running these services is available as part of our `motools` project[11].

### 6.3.1 Magnatune

Magnatune is a music label providing its content under a Creative Commons non-commercial license. It holds around 250 artists, 500 albums and 9,000 tracks. Magnatune provides a SQL dump of their internal database.

We first used the D2R software described in § 6.2.1, and a corresponding D2R mapping to publish Music Ontology data out of the Magnatune database. Due to restrictions in the form of the web identifiers generated by D2R, we set up an instance of the UriSpace software described in § 6.2.3 on top of the D2R SPARQL end-point.

For example, the following web resource identifies the "American Baroque" chamber music quartet.
`http://dbtune.org/magnatune/artist/american_baroque`
The associated structured representation is available at the following web identifier.
`http://dbtune.org/magnatune/sparql/?query=describe%20%3Chttp://dbtune.org/magnatune/artist/american_baroque%3E`
This web identifier corresponds to the following query on the Magnatune SPARQL end-point.

```
DESCRIBE <http://dbtune.org/magnatune/artist/american_baroque>
```

We designed the corresponding structured representation using the Music Ontology terms defined in § 3.3.1.

```
<http://dbtune.org/magnatune/artist/american_baroque>
    a mo:MusicArtist;
    dc:description """Spectacular Baroque and
Classical chamber music"""^^xsd:string;
    foaf:based_near <http://dbpedia.org/resource/USA>;
    foaf:homepage <http://magnatune.com/artists/american_baroque>;
    foaf:img <http://magnatune.com//artists/img/american_baroque_1.jpg>;
    foaf:name "American Baroque"^^xsd:string .



<http://dbtune.org/magnatune/performance/828>
    mo:performer <http://dbtune.org/magnatune/artist/american_baroque> .

<http://dbtune.org/magnatune/track/775>
    foaf:maker <http://dbtune.org/magnatune/artist/american_baroque> .

<http://dbtune.org/magnatune/album/abaroque-rameau>
```

[11] http://sourceforge.net/projects/motools/

```
    foaf:maker <http://dbtune.org/magnatune/artist/american_baroque> .

[...]
```

The Magnatune linked data and SPARQL end-point are available at [Rai07e]

### 6.3.2 Jamendo

Jamendo is a web music platform. Artists can upload albums and choose a Creative Commons license or a Free Art License to apply to their content. Then, Jamendo members can make donations to these artists, download the records, rate them, tag them, etc. Jamendo holds around 10,000 albums. Jamendo provides a REST web service and an XML dump of their internal database.

We used our P2R software described in § 6.2.2 to publish this dataset, using the Jamendo web service as an input. A SWI-Prolog module wraps calls to the Jamendo web service in a number of Prolog predicates. We wrote a P2R mapping for translating these predicates to RDF. On top of that, an instance of our UriSpace software maps web resources to corresponding structured representations generated by the P2R SPARQL end-point.

For example, the following web resource corresponds to the French band "Both".
`http://dbtune.org/jamendo/artist/5`
The corresponding structured representation is, as in the Magnatune dataset described above, generated by a DESCRIBE query on the P2R SPARQL end-point. This representation is as follows.

```
<http://dbtune.org/jamendo/artist/5>
  a mo:MusicGroup;
  foaf:made <http://dbtune.org/jamendo/record/174>;
  foaf:made <http://dbtune.org/jamendo/record/33>;
  owl:sameAs <http://dbtune.org/musicbrainz/resource/artist/
0781a3f3-645c-45d1-a84f-76b4e4decf6d>;
  foaf:based_near <http://sws.geonames.org/2991627/>;
  foaf:homepage <http://www.both-world.com>;
  foaf:img <http://img.jamendo.com/artists/b/both.jpg>;
  foaf:name "Both"^^xsd:string.
```

We also use UriSpace to expose a number of informational documents. For example, the following web resource gives access to all instances of an artist in the Jamendo dataset.
`http://dbtune.org/jamendo/all/artist`
The Jamendo linked data and SPARQL end-point are available at [Rai07d].

### 6.3.3 BBC John Peel sessions

John Peel was an influential disk jockey on BBC Radio 1, from 1967 to 2004. The BBC released in 2007 a dataset describing all the John Peel sessions[12] under a Creative Commons license. This

[12]Several artists were recorded in a studio specifically for broadcast on John Peel's BBC radio 1 show. Usually, a session consists of four performances.

dataset covers 2,274 artists and 3,976 sessions and is available in a tab-separated values (TSV) format.

We also used our P2R and UriSpace tools to publish this dataset. We wrote a SWI-Prolog module wrapping the access to the TSV files. Then, a P2R mapping translates the corresponding Prolog predicates to Music Ontology RDF. Here, we mainly used the terms defined in § 3.3.2. A single session corresponds to a particular performance. This complex performance is divided into several simpler performances, as described in § 3.3.3.3. These simpler performances correspond to the different musical works being performed, and the instruments played by the different performers. We also describe a recording event corresponding to the recording of the whole performance, to which we attach sound engineers and producers. We reuse the broadcast concept defined in the Programmes Ontology [SRSH08], subsuming the event concept defined in § 3.2.2.2. Different broadcasts of a single session are then also part of the dataset.

For example, the following web resource corresponds to the band "L7".
`http://dbtune.org/bbc/peel/artist/1049`
This web resource gives access to the following structured representation, corresponding to a DESCRIBE query on the end-point[13].

```
<http://dbtune.org/bbc/peel/artist/1049>
  a mo:MusicArtist;
  mo:performed <http://dbtune.org/bbc/peel/session/1804>;
  rdfs:label "L7"^^xsd:string;
  owl:sameAs <http://dbpedia.org/resource/L7_(band)>;
  foaf:img <http://bbc.co.uk/music/>;
  foaf:name "L7"^^xsd:string .

<http://dbtune.org/bbc/peel/session/1804>
  a mo:Performance;
  event:place [rdfs:label "Maida Vale 5"];
  event:sub_event
 <http://dbtune.org/bbc/peel/perf_ins/22a180124e1c830485b86a060a1125be>,
 <http://dbtune.org/bbc/peel/perf_work/13122>,
  mo:performer <http://dbtune.org/bbc/peel/artist/1049>;
  mo:producesSound <http://dbtune.org/bbc/peel/sound/1804>;
  rdfs:label "Performance 1804 in Maida Vale 5"^^xsd:string .

<http://dbtune.org/bbc/peel/perf_ins/22a180124e1c830485b86a060a1125be>
  a mo:Performance;
  mo:instrument "Bass, Vocals";
  mo:performer
   <http://dbtune.org/bbc/peel/artist/22a180124e1c830485b86a060a1125be> .

<http://dbtune.org/bbc/peel/artist/22a180124e1c830485b86a060a1125be>
  a foaf:Person;
```

[13]We here concatenate the representation of neighbouring resources, in order to give an idea of the RDF information available.

```
  foaf:name "Jennifer Finch"^^xsd:string .

<http://dbtune.org/bbc/peel/perf_work/13122>
  a mo:Performance;
  mo:usesWork <http://dbtune.org/bbc/peel/work/13122> .

<http://dbtune.org/bbc/peel/work/13122>
  a mo:MusicalWork;
  dc:title "Let's Lynch The Landlord"^^xsd:string;
  rdfs:label "Let's Lynch The Landlord"^^xsd:string .
```

The BBC John Peel linked data and SPARQL end-point are available at [Rai07c].

### 6.3.4 Audioscrobbler data

The AudioScrobbler system [aud] tracks listening habits. A number of plug-ins are available for a wide range of music players, which submit information about the tracks being played to AudioScrobbler. Audioscrobbler provides XML web services. These web services corresponds to XML documents served at a particular web identifier, and exposing a specific part of the AudioScrobbler database. For example, the following web identifier gives access to some information (title, artist, record, play time) about the last tracks played by the user `moustaki`.

`http://ws.audioscrobbler.com/1.0/user/moustaki/recenttracks.xml`

We wrapped the following XML web services in a linked data interface.

- Relationships between users;
- Recently listened tracks;
- Recommended events.

Again, we used our P2R and UriSpace softwares to publish web identifiers and corresponding representations, derived from the output of these web services. We had to define a small extension of the Music Ontology framework to cover a new range of events, corresponding to a person listening to a particular track. Hence, we define a new 'ScrobbleEvent' concept, involving a listener and a track at a particular place and time.

The following web resource corresponds to a particular user of the AudioScrobbler service.
`http://dbtune.org/last-fm/moustaki`
This web resource gives access to the following structured representations, generated from the output of the web services mentioned above.

```
<http://dbtune.org/last-fm/moustaki>
    a foaf:Person;
    rdfs:label "moustaki";
    foaf:knows <http://dbtune.org/last-fm/cygri>,
               <http://dbtune.org/last-fm/njh>;
    foaf:holdsAccount [
        a foaf:OnlineAccount;
        foaf:accountName "moustaki";
        foaf:accountServiceHomepage <http://www.last.fm/>;
        foaf:primaryTopicOf <http://www.last.fm/user/moustaki>
    ] .

[]
    a lfm:ScrobbleEvent;
    rdfs:label "Listened to All, track Million Bucks, record Pummel";
    dc:date "2008-05-28T12:53:55.0Z"^^xsd:dateTime;
    lfm:track_played _:bn8;
    lfm:user <http://dbtune.org/last-fm/moustaki> .

_:bn8
    a mo:Track;
    dc:title "Million Bucks";
    rdfs:label "Million Bucks";
    foaf:maker [
        a mo:MusicalArtist;
        foaf:name "All";
        owl:sameAs <http://dbtune.org/musicbrainz/resource/artist/
e92547b5-a134-4149-af9a-c944af9e476f>
    ];
    foaf:primaryTopicOf <http://www.last.fm/music/All/_/Million%2BBucks> .

[]
    a mo:Record;
    rdfs:label "Pummel";
    foaf:name "Pummel";
    mo:track _:bn8;
    owl:sameAs <http://dbtune.org/musicbrainz/resource/record/
10800e0a-9057-40e6-9828-300b7ce5e1c1> .
```

The Audioscrobbler linked data is available at [Rai07b].

### 6.3.5 Musicbrainz

Musicbrainz is a community-maintained music database holding detailed editorial information about 300,000 artists, 500,000 releases and 6 million tracks. A database schema was developed to encompass a wide range of editorial information. This schema can be depicted as in Figure 6.4. A PostgreSQL database dump is made available by the Musicbrainz community.

The schema evolved from the main tables depicted in Figure 6.4. The "Advanced Relationships" mechanism was added to extend the expressiveness of the initial schema (relationships

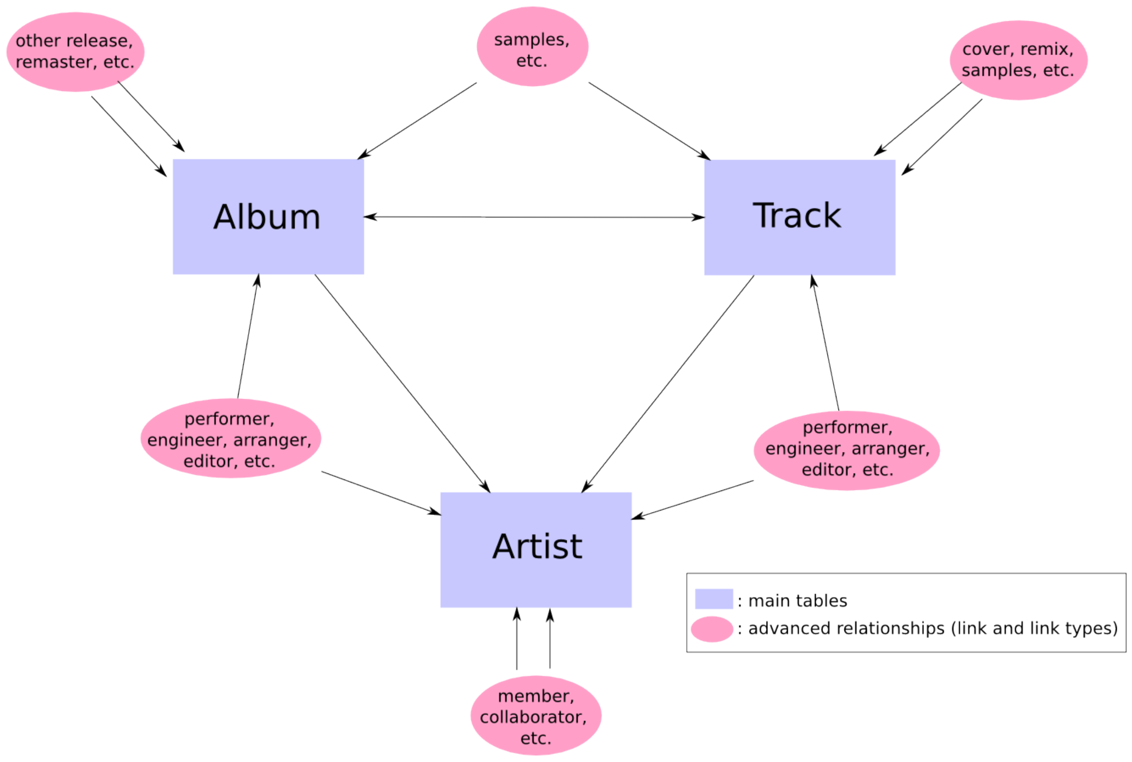

other release, remaster, etc.
samples, etc.
cover, remix, samples, etc.
Album
Track
performer, engineer, arranger, editor, etc.
performer, engineer, arranger, editor, etc.
Artist
: main tables
: advanced relationships (link and link types)
member, collaborator, etc.

Figure 6.4: Depiction of the Musicbrainz database schema

between artists, between different tracks, etc.). However, the current schema has many limitations, especially for the handling of classical music. Describing multiple performances of a particular work is impossible. These limitations are being addressed in a new schema based on concepts similar to the concepts defined in § 3.3.2.

We developed a D2RQ mapping to make the Musicbrainz relational database available as linked data and through a SPARQL end-point. We manually mapped the Musicbrainz advanced relationships to corresponding concepts in the Music Ontology. For example, the advanced relationship "performer" along with the corresponding instrument between a track and an artist would be related to an instance of the performance concept in the Music Ontology. However, information pertaining to the musical work is difficult to map. This information is indeed encoded within the title of a particular track, using some capitalisation rules defined by the Musicbrainz community. Such information is therefore not queriable in the current Musicbrainz system. The new schema should address this issue.

The following web resource identifies Glenn Gould performing the second book of the Johann Sebastian Bach's Well Tempered Clavier.
`http://dbtune.org/musicbrainz/resource/performance/38641`
This web resource gives access to the following structured representation, generated from a PostgreSQL database holding the Musicbrainz database dump.

```
<http://dbtune.org/musicbrainz/resource/performance/38641>
  a mo:Performance ;
  rdfs:label "Glenn Gould performing (recorded on album The Well-Tempered
Clavier, Book II (feat. piano: Glenn Gould) (disc 1))" ;
  mo:instrument <http://dbtune.org/musicbrainz/resource/instrument/180> ;
  mo:performer <http://dbtune.org/musicbrainz/resource/artist/
7002bf88-1269-4965-a772-4ba1e7a91eaa> ;
  mo:recorded_as <http://dbtune.org/musicbrainz/resource/record/
b4592c03-ab70-40ce-9227-f9726fe0f797> .

<http://dbtune.org/musicbrainz/resource/artist/7002bf88-1269-4965-a772-
4ba1e7a91eaa>
  a mo:MusicArtist ;
  rdfs:label "Glenn Gould" ;
  bio:event
    <http://dbtune.org/musicbrainz/resource/artist/7002bf88-1269-4965-a772-
4ba1e7a91eaa/birth>,
    <http://dbtune.org/musicbrainz/resource/artist/7002bf88-1269-4965-a772-
4ba1e7a91eaa/death> ;
  foaf:homepage <http://www.glenngould.com> ;
  owl:sameAs <http://dbpedia.org/resource/Glenn_Gould> .

<http://dbtune.org/musicbrainz/resource/instrument/180>
  a owl:Class ;
  rdfs:label "Piano" ;
  rdfs:subClassOf <http://dbtune.org/musicbrainz/resource/instrument/322> .
```

The Musicbrainz linked data and SPARQL end-point are available at [Rai08b].

### 6.3.6 BBC Playcount data

The BBC released in June 2008 a dataset of playcount data, linking BBC programmes to artists featured in them. For example, this dataset holds such information as "Tom Waits was featured 3 times on the BBC radio 6 Chris Hawkins show". We converted the original data to RDF, linking Musicbrainz artists to BBC brands within BBC Programmes[14]. We serve the resulting RDF using the SWI Prolog Semantic Web server [WHM08] along with our UriSpace module.

For example, the following web resource identifies the playcount of Tom Waits on the BBC Chris Hawkins show.

`http://dbtune.org/bbc/playcount/b00721l8y_952`

This web resource gives access to the following structured data, linking a Musicbrainz artist resource in the Semantic Web publication described in § 6.3.5 and a brand resource in BBC Programmes.

```
<http://www.bbc.co.uk/programmes/b00721l8y#brand>
    pc:playcount <http://dbtune.org/bbc/playcount/b00721l8y_952> .

<http://dbtune.org/bbc/playcount/b00721l8y_952>
    a pc:Playcount;
    pc:count "3"^^xsd:int;
```

[14]BBC Programmes is a large web repository of information about every brands, series, episodes broadcasted on any BBC service. It is available at `http://www.bbc.co.uk/programmes/`.

```
    pc:object <http://dbtune.org/musicbrainz/resource/artist/
c3aeb863-7b26-4388-94e8-5a240f2be21b> .
```

The BBC Playcount linked data and SPARQL end-point are available at [Rai08a].

### 6.3.7 Chord service

We developed a service providing web identifiers for musical chords, as mentioned in § 3.3.4.2. We used Harte's chord syntax [HSAG05] as a starting point, and derived web identifier from that. For example, the chord written as `Ds:min7(*b3,9)/5` (a D sharp minor with added ninth and missing flat third, over the fifth, depicted in Figure 3.7) in Harte's syntax corresponds to the following web identifier.

```
http://purl.org/ontology/chord/symbol/Ds:min7(*b3,9)/5
```

The structured representations associated to these chord identifiers link to the Chord ontology terms defined in § 3.3.4.2. This representation, generated from parsing Harte's syntax in the suffix of the web identifier, is as follows.

```
<http://purl.org/ontology/chord/symbol/Ds:min7(*b3,9)/5>
    a chord:Chord;
    chord:interval [
        a chord:ScaleInterval;
        chord:degree "9"^^xsd:int
    ], [
        a chord:ScaleInterval;
        chord:degree "1"^^xsd:int
    ], [
        a chord:ScaleInterval;
        chord:degree "5"^^xsd:int
    ], [
        a chord:ScaleInterval;
        chord:degree "7"^^xsd:int;
        chord:modifier chord:flat
    ];
    chord:base_chord chord:min7;
    chord:bass [
        a chord:ScaleInterval;
        chord:degree "5"^^xsd:int
    ];
    chord:root [
        a chord:Note;
        chord:modifier chord:sharp;
        chord:natural <http://purl.org/ontology/chord/note/D>
    ];
    chord:without_interval [
        a chord:ScaleInterval;
        chord:degree "3"^^xsd:int;
        chord:modifier chord:flat
    ];
    foaf:depiction <http://rvw.doc.gold.ac.uk/omras2/widgets/chord/
Ds:min7(*b3%2C9)/5> .
```

When transcribing the chords in a particular musical work, we can just refer to these web identifiers for chords. Accessing one of these identifiers leads to a structured representation of the corresponding chord. A large dataset of such transcriptions is available at [cho]. The chord service is available at [Rai07g].

### 6.3.8 Summary

Overall, the DBTune server gives access to eight distinct linked data sources[15]. It is at the time of writing one of the largest information sources available on the Semantic Web, with more than 15 billion RDF triples available. We will quantify some of its characteristics in § 6.5. The published information covers social networks, listening habits, detailed editorial data, chords and Creative Commons music repositories.

Other services are giving access to Music Ontology information. For example, the BBC Music web site[16] provides Music Ontology RDF. A GRDDL transformation[17] translates hAudio Microformats to Music Ontology RDF, as described in § 2.2.4.4. The Benjamin Britten Thematic Catalogue[18] is available as Music Ontology RDF. A service also enables the conversion of ID3 tags embedded within audio files to Music Ontology RDF[19].

## 6.4 Interlinking real-world datasets

In the previous section, we described the datasets we published as sets of web identifiers and corresponding structured representation. A number of other efforts have published datasets in other domains, as pointed out in § 6.1. These include DBpedia [ABL+07], publishing structured data extracted from the Wikipedia online encyclopedia, and Geonames. In the previous examples of RDF representations, we saw links from one dataset to another. For example, in § 6.3.5, the RDF code snippet includes an `owl:sameAs` statement linking Glenn Gould in Musicbrainz to Glenn Gould in DBpedia. In this section, we describe and evaluate different techniques for drawing links from one web information source to another.

[15]The eighth source is a linked data wrapper around the MySpace social network.
[16]available at `http://www.bbc.co.uk/music/beta/`
[17]available at `http://weborganics.co.uk/mo-haudio/`
[18]available at `http://musariada.mus.uea.ac.uk/btc/`
[19]available at `http://mtg100.upf.es/foafing-the-music-news/?p=18`

### 6.4.1 Manual interlinking

A large number of the links depicted in Figure 6.2 are asserted manually. For example, in FOAF, each individual maintains its own set of links, relating them to e.g. acquaintances, interests, pictures. The links between Musicbrainz and DBpedia were drawn manually by the Musicbrainz community. Indeed, one part of the advanced relationships described in § 6.3.5 allows the community to link artists, records or tracks to corresponding web identifiers on other web sites. The user-contributed interlinking strategy [HHR08] specifies this approach in a domain-independent context. It defines a simple web application allowing any consumer of a particular web identifier to relate it to further web identifiers, through a range of RDF properties (`owl:sameAs`, but also `rdfs:seeAlso` or `foaf:isPrimaryTopicOf`).

### 6.4.2 Towards automatic interlinking

Manually interlinking large datasets, especially those that are not easily editable by the community, is impractical. We need a way to automatically detect the overlapping parts of heterogeneous datasets. Such an approach is proposed in this subsection.

#### 6.4.2.1 Definition of the problem

We define the automatic interlinking problem as follows. We consider two web datasets $A$ and $B$, respectively describing a number of web resources $a$ and $b$. We consider the problem of finding resources $b$ in our dataset $B$, which identify the same object as a resource $a$ in our dataset $A$. For example, we want to find that the band named 'Both' in Jamendo is the same as a particular band named 'Both' in Musicbrainz. Our problem is then automatic co-referencing, i.e. finding out when two identifiers refer exactly to the same entity[20]. We want to achieve automatic co-referencing in a linked data context, so that the datasets described in § 6.3 and others on the Web can be automatically interlinked.

In such a context, we need to consider two main issues. On the one hand, we must ensure that we do not publish an equivalency statement between two web resources that correspond to two different entities—we must have a low rate of false-positives. For example, we do not want to draw an equivalency link between a resource identifying a musical work entitled "Violet" and a flower, which would lead to wrong inferences on the user agent side (e.g. "the song Violet has a violet colour"). The inverse error (not publishing an equivalency statement between two resources identifying the same entity) is less harmful. On the other hand, we do not want to have multiple possible interlinking decisions for one resource—we want to limit the amount of manual post-processing.

Also, we do not consider the problem of ontology matching [ES07]. That is, we consider that the different datasets we want to interlink use similar ontologies.

[20]All mapping problems in our context can be reduced to this problem, even *literal expansion* where a resource is linked to a literal and we are looking for the corresponding web resource, e.g. expanding "Moselle, France" into the corresponding web identifier in a geographic dataset. In these cases, a simple transformation in the dataset $A$ is enough to return ourselves to the above defined problem. For example, the triple `:both foaf:based_near ''Moselle, France''` is transformed into `:both foaf:based_near :geo. :geo rdfs:label ''Moselle, France''`. Then, the problem is to match the `:geo` resource to the corresponding resource in the geographic dataset.

#### 6.4.2.2 Background

Our problem is similar to the problem of record linkage [Win99] in the database community. Record linkage consists of finding out when two records or files in different databases represent identical entities. Fellegi and Sunter [FS69] give a mathematical model of record linkage. They define two sets of entities $A$ and $B$. Some elements are common to these two sets. The Cartesian product $A * B$ is then composed of two disjoint sets: $M = (a, b); a = b, a \in A, b \in B$ and $U = (a, b); a \neq b, a \in A, b \in B$. Each entity $a \in A$ and $b \in B$ has a set of characteristics respectively denoted by $\alpha(a)$ and $\beta(b)$. When performing a linkage operation, we observe a *comparison vector* $\gamma[\alpha(a), \beta(b)]$. This comparison vector is a set of statements such as "the name is the same", "the title is the same, but the maker is different", etc. From this set of statements, we want to take one of the following decisions. A *positive link* decision, denoted by $A_1$, means that $(a, b) \in M$. A *positive non-link* decision, denoted by $A_3$, means that $(a, b) \in U$. A *possible link* decision, denoted by $A_2$, corresponds to cases in which we are not able to make a decision. Our goal is then to find a *linkage rule*, defined as a mapping from the set of possible realisations of $\gamma$ to a set $d(\gamma)$ of random decision functions. If $P(A_i|\gamma), i \in \{1, 2, 3\}$ is the conditional probability of taking the $A_i$ decision given $\gamma$, we can write $d(\gamma)$ as follows.

$$d(\gamma) = \{P(A_1|\gamma), P(A_2|\gamma), P(A_3|\gamma)\} \text{ and } \sum_{i \in \{1,2,3\}} P(A_i|\gamma) = 1 \tag{6.1}$$

The theory also considers the two types of error mentioned in § 6.4.2.1: taking a positive link decision when $(a, b) \in U$, and taking a positive non-link decision when $(a, b) \in M$. The *optimal linkage rule* maximises the probabilities of $A_1$ and $A_3$. The amount of manual post-processing to be made is minimised.

Jaffri et al. [JGM07] argue that this theory can be used for identifying probabilities of having web resources identifying the same entity. Indeed, we can consider the structured representation of a web resource $a$ as a set of characteristics $\alpha(a)$, with different fields corresponding to the different RDF literals attached to this resource.

However, we argue in the § 6.4.3 and in § 6.4.4 that such a methodology is only applicable in specific cases. This methodology generally leads to poor interlinking results. We need to consider the existence of further resources, linked from the original one, that give further clues to support the interlinking decision. We show that we need to not only match corresponding web resources, but to match corresponding graphs of web resources. We present in § 6.4.5 our algorithm to do so.

### 6.4.3 Literal-based interlinking

Directly applying the record linkage theory to structured web data amounts to deriving a comparison vector $\gamma[\alpha(a), \beta(b)]$ for two web resources $a$ and $b$, that holds the result of a comparison between matching literal properties. For example, if we have the predicates $\textsf{foaf:name}(a, l_1)$ and $\textsf{foaf:name}(b, l_2)$, one element of the comparison vector will correspond to a comparison between the two literals $l_1$ and $l_2$.

This simple approach can be suitable for interlinking datasets in cases where a literal string reliably provides sufficient disambiguation. We used this approach to automatically interlink the Jamendo dataset described in § 6.3.2 and the Geonames dataset, as the Jamendo community takes special care of having literals that are disambiguating enough for geolocation information, and that the Geonames dataset handles only that particular kind of information. We use the Geonames search web service to return the Geonames resource corresponding to a particular literal. We evaluate this interlinking over 60 random geographic locations in the Jamendo dataset—20% of the geographic locations available in this dataset. We manually checked the accuracy of such an interlinking methodology. For these 60 locations, 60 positive link decisions were taken, and all were correct.

We applied this methodology to try mapping the BBC John Peel sessions dataset described in § 6.3.3 and DBpedia [ABL+07]. The resources we tried to map were identifying musicians, engineers, producers and musical works.

For example, we consider the following resource in the BBC John Peel dataset, corresponding to a song by the band "Hole", entitled "Violet".

```
<http://dbtune.org/bbc/peel/work/1498>
    a mo:MusicalWork;
    rdfs:label "Violet"^^xsd:string .
```

This resource has one literal attached to it, through the `rdfs:label` property. Looking for resources in the DBpedia dataset that share such a property leads to sixteen results. No interlinking decision can be taken, as we can only assess that there is a possible link between our "Violet" resource and each of these sixteen DBpedia resources. Such an interlinking therefore involves a significant amount of post-processing, which goes against one of the requirements in § 6.4.2.1.

### 6.4.4 Constrained literal-based interlinking

A solution is to add constraints on the resulting resources. We add information about matching types (objects of a `rdf:type` predicate) to our comparison vector $\gamma$. For example, by using the DBpedia links to Yago [SKW07], we can restrict DBpedia resources to be of a specific Yago type. For our "Violet" example, the following query on DBpedia gives one result—the actual song.

```
SELECT ?r
WHERE {
?r p:name "Violet"@en .
?r a <http://dbpedia.org/class/yago/Song107048000> }
```

This query constrains the type of the resource we are looking for. We used this approach to map the BBC John Peel sessions to corresponding resources in DBpedia. The correspondences between Music Ontology and Yago concepts were manually defined. Our comparison vector $\gamma[\alpha(a), \beta(b)]$ includes elements encoding the similarity of literals and an element encoding the similarity of types. In our example, the literal similarity is encoded by "the strings are the same" or "the strings are not the same". The type similarity is encoded by "the types are matching" or "the types are not matching". If for two resources $a$ and $b$ the types are matching and literal strings corresponding to similar properties are the same, we take a positive link decision. Otherwise, we take a positive non-link decision.

| | | Positive link | Positive non-link |
|---|---|---|---|
| Person resources | Correct | 26.66% | 40% |
| | Incorrect | 1.66% | 31.66% |
| Musical works | Correct | 14% | 42% |
| | Incorrect | 36% | 8% |

Table 6.1: Evaluation of constrained literal-based interlinking on person and musical work resources within DBpedia and the BBC John Peel sessions

To evaluate this interlinking, we consider 60 random persons (2.6% of the overall number of persons in the BBC John Peel dataset). We manually check the correctness of the interlinking. We classify the interlinking results in the following categories.

- Correct positive link;
- Incorrect positive link;
- Correct positive non-link;
- Incorrect positive non-link.

For our person test set, the results are displayed in table 6.1. The most common error made by the constrained literal-based interlinking algorithm is to take a positive non-link decision although a matching resource exists in DBpedia. The main reason for these errors is that we consider only two possibilities when matching two literals. They are either exactly the same, or they do not match. However, many variations can exist for an artist name. Some can be captured by an appropriate string similarity metric, such as the difference between "Boys of the Lough" and "The Boys of the Lough". Some other variations are more difficult to capture, such as the variation between "Buckfunk" and "Si Begg", where one string is an alias for the artist, and the other string is a stage name. We therefore need a better way to handle similarities between literals attached to a particular resource. Apart from these positive non-link errors, the interlinking results are mostly correct, except marginal cases where two artists have the same name.

We repeated a similar evaluation for 50 musical works within the BBC John Peel session dataset—0.2% of the musical works available in this dataset. The overlap between the two datasets is low: of one hundred musical work resources in the BBC John Peel sessions, 8 were also in DBpedia. We therefore adapted our testset. Half of it are resources for which our algorithm took a positive link decision, and half of it are resources for which our algorithm took a positive non-link decision. The results are displayed in table 6.1. 16% of the positive non-link decision made by the algorithm were wrong, and 72% of the positive link decisions were wrong. This goes clearly against one of the requirements in § 6.4.2.1.

Such bad results are mainly due to two factors. The first factor is the actual structure of DBpedia, which does not match exactly our concept of a musical work. DBpedia has concepts corresponding to "singles" (published version of a particular performance of that work), "albums", or "songs". These concepts often clash—a single can be considered as being the same as a song, for example. The second factor is that, even with restrictions on the nature of the target resource taking into account these ontology clashes, associated literals may not be discriminating enough. For example, there exists two songs entitled "Mad Dog" in DBpedia, one by Elastica and one by Deep Purple. Even within the BBC John Peel session dataset, the title of a song is not

| Resource 1 | Resource 2 | Initial Similarity |
|---|---|---|
| <u>1</u> | <u>4</u> | 1 |
| <u>1</u> | <u>7</u> | 1 |
| <u>2</u> | <u>5</u> | 0.933 |
| <u>3</u> | <u>6</u> | 0.951 |
| <u>2</u> | <u>6</u> | 0.17 |
| <u>3</u> | <u>5</u> | 0.182 |
| <u>2</u> | <u>8</u> | 0.27 |
| <u>3</u> | <u>8</u> | 0.314 |

Table 6.2: Initial resource similarities

discriminating enough. There are four distinct works entitled "Four", for example. We came across the same problem in the interlinking of persons, although in our particular scenario, artist names tend to be disambiguating enough. Other datasets may have several artists associated with a single name. For example, there are two distinct bands named "Both" in Musicbrainz. An intuitive approach to disambiguate between the two artists would be to check the titles of their releases. If by any chance we are still not able to take a decision, we can check the titles of the corresponding tracks. We need to take into account the similarities of related resources.

This evaluation of constrained literal-based interlinking leads us to the two following requirements, for an efficient automated interlinking algorithm:

- Handle similarities between literals;
- Handle similarities of related resources.

### 6.4.5 Graph-based interlinking

There is a need for a more sophisticated interlinking algorithm handling these two points. We describe here an algorithm exploring structured web data when an interlinking decision cannot be made. The underlying data model then moves from flat database records as in § 6.4.2.2 to a graph-based structure, modelling the inter-dependencies between related web resources. The graph-based interlinking algorithm will output interlinking decisions for a whole graph of web resources. For example, in the case of an editorial dataset, it will output at once a matching artist, the corresponding matching records and the matching tracks on these records.

#### 6.4.5.1 Offline graph mapping

We consider here that we have access to two datasets $A$ and $B$, which we want to interlink. To illustrate our graph-based interlinking algorithm, we consider the two datasets illustrated in Figure 6.5, with our dataset on the left containing a single artist with the name "Both", and the dataset on the right containing two artists named "Both". We model the two datasets as graphs, with each edge representing an RDF triple $(s, p, o)$.

We first compute initial similarity values between all pairs of resources $(s_1, s_2)$ and $(o_1, o_2)$ such that $(s_1, p, o_1) \in A$ and $(s_2, p, o_2) \in B$. Such similarity values can be calculated by a string similarity algorithm comparing literals directly attached to these resources, using for example the Damerau-Levenshtein distance measures [Dam64, Lev66] or the Jaro string comparator [Jar89]. In our example, a string similarity based on a minimal edit distance produces the results in table 6.2.

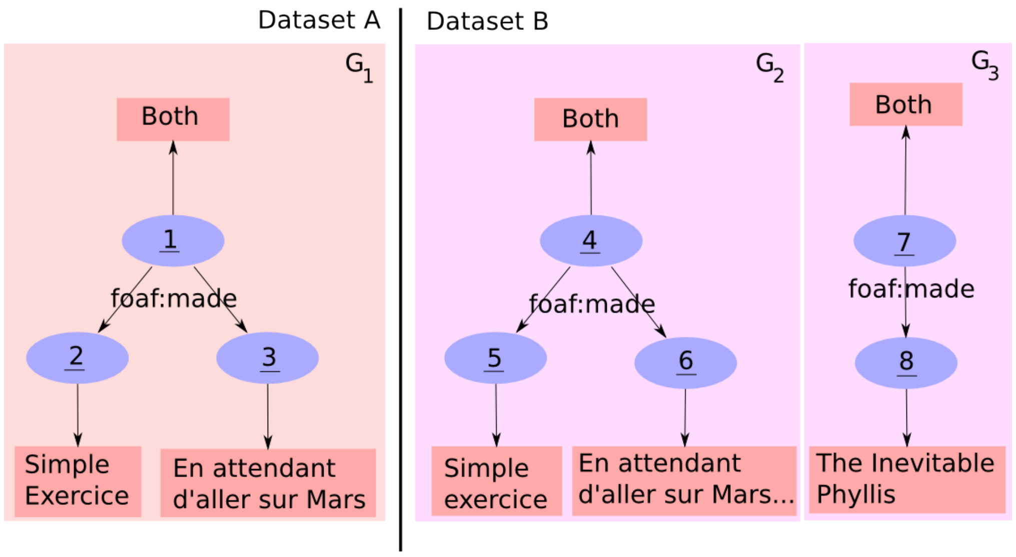


Figure 6.5: Example datasets—we here try to match elements from A and B

| Graphs | | Mapping | Measure |
|---|---|---|---|
| $G_1$ | $G_2$ | $M_{G_1:G_2}a = \{(\underline{1}, \underline{4}), (\underline{2}, \underline{5}), (\underline{3}, \underline{6})\}$ | 0.961 |
| $G_1$ | $G_2$ | $M_{G_1:G_2}b = \{(\underline{1}, \underline{4}), (\underline{2}, \underline{6}), (\underline{3}, \underline{5})\}$ | 0.451 |
| $G_1$ | $G_3$ | $M_{G_1:G_3}a = \{(\underline{1}, \underline{7}), (\underline{2}, \underline{8})\}$ | 0.635 |
| $G_1$ | $G_3$ | $M_{G_1:G_3}b = \{(\underline{1}, \underline{7}), (\underline{3}, \underline{8})\}$ | 0.657 |

Table 6.3: Possible graph mappings and associated measures

We now construct possible graph mappings between sub-graphs of $A$ and sub-graphs of $B$. If $G$ is a sub-graph of $A$ and $H$ is a sub-graph of $B$, a possible graph mapping between $G$ and $H$ is a set of tuples $(g, h)$ mapping one node of $G$ to one node of $H$. We associate to such a graph mapping $M$ a measure $\mathsf{measure}(M)$, capturing how costly it is to apply the associated correspondences. To derive this measure, we sum the similarity values $\mathsf{sim}(g, h)$ associated with each pair of resources $(g, h)$ in the graph mapping. We normalise the resulting value by the number of pairs in the mapping.

$$\mathsf{measure}(M) = \frac{1}{|M|} * \sum_{(g,h)\in M} \mathsf{sim}(g, h) \tag{6.2}$$

where $\mathsf{sim}(g, h)$ is the literal-based similarity value between two resources $g$ and $h$.

For the example depicted in Figure 6.5, we consider the possible graph mappings in table 6.3, along with the corresponding resulting measures. These measures are computed from the resource similarities in table 6.2.

Now, we can choose the mapping whose measure is the highest, optionally thresholding to avoid mapping graphs that are too dissimilar. In our example, we choose $M_{G1:G2}a$.

### 6.4.5.2 Online graph mapping

We now formalise this graph-based interlinking algorithm in an online context. We do not know exactly the entire structure of $A$ and $B$ when starting our algorithm. Instead, we discover structured web data as we go, and update our local image of $A$ and $B$ accordingly, as well as the possible graph mappings and their measures. This is necessary because in general the size of the datasets $A$ and $B$ will be prohibitive; if not for loading both datasets, then for computing all the possible graph mappings between them.

Our starting point is now a single URI $a$ in a dataset $A$. Our objective is to find the corresponding $b$ in a dataset $B$, as well as the correspondences between resources in the neighbourhood of $a$ and resources in the neighbourhood of $b$.

In the following, we will use the Named Graphs approach described in § 2.2.4.2 to associate a web resource with the graph retrieved when accessing a RDF representation of it, and we let $G_x$ denote this graph for a resource $x$.

Our graph-based interlinking algorithm consists of the following steps.

1. Initially, we extract suitable labels $l$ for $a$ in $G_a$, using properties such as `dc:title` or `foaf:name`.

2. We now need to access some potential candidates for mapping in $B$. To do that, we use the same approach as in § 6.4.4. We issue a query to $B$ which involves $l$ and constraints over what we are looking for, through a SPARQL end-point or a custom web service[21]. This gives us a list of resources.

3. For each $b_k$ in this list, we access $G_{b_k}$. Now, for all possible graph mappings $M_{G_a:G_{b_k},i}$, we compute the measure defined in (6.2).

4. If we can make a clear decision now (i.e. we have one unique mapping for which the measure is above our decision threshold), we terminate and choose the corresponding graph mapping.

5. If not, we look for object properties $p$ such that $(a, p, o) \in G_a$ and $(b_k, p, o') \in G_{b_k}$, and we access $G_o$ and $G_{o'}$.[22] We update our possible graph mappings and the associated measures.

6. We go back to step 4.[23]

We give the full pseudo-code of our graph-based interlinking algorithm in Appendix C.

#### 6.4.5.3 Example

We illustrate this algorithm starting from `http://dbtune.org/jamendo/artist/5`, a web resource identifying an artist in the Jamendo dataset. We use $\underline{1}$ as a short symbol for this web identifier. We try to map this resource to the corresponding web identifier in the Musicbrainz dataset. As part of the same process we wish to map corresponding albums and tracks for this artist.

We access in $G_{\underline{1}}$ that our starting resource is called "Both". We now look for a band named "Both" in the Musicbrainz dataset, through the Musicbrainz web service[24]. This gives us back two web identifiers corresponding to two bands named "Both": $\underline{4}$ and $\underline{7}$. We now consider two possible graph mappings with two measures, both equal to one. We continue looking for further clues, as there is not yet any way to disambiguate between the two.

We access in $G_{\underline{1}}$ that $\underline{1}$ made[25] two things: $\underline{2}$ and $\underline{3}$. We access in $G_{\underline{4}}$ that $\underline{4}$ made two things: $\underline{5}$ and $\underline{6}$. We access in $G_{\underline{7}}$ that $\underline{7}$ made one thing: $\underline{8}$. As this property is occurring in these three

[21]such as `http://sindice.com/`

[22]We could additionally consider triples of the form $(o, p, a)$ and $(o', p, b_k)$.

[23]Practically, we also limit the maximum number of iterations the algorithm may perform.

[24]`http://wiki.musicbrainz.org/XMLWebService`

[25]captured through the `foaf:made` predicate

| | Positive link | Positive non-link |
|---|---|---|
| Correct | 8.33% | 88.33% |
| Incorrect | 0% | 3.33% |

Table 6.4: Evaluation of the Jamendo/Musicbrainz interlinking using our graph-based interlinking algorithm

graphs, we access more information about its objects. Our starting band 1 made two records, entitled “Simple Exercice” and “En attendant d’aller sur Mars”. The Musicbrainz band 4 made two records, entitled “Simple exercice” and “En attendant d’aller sur Mars...”. The Musicbrainz band 7 made one record, entitled “The Inevitable Phyllis”. We now update the possible graph mappings, and reach the results in table 6.3. Now, we have one mapping identifiably better than the others, with a measure of 0.961. This graph mapping corresponds to the following statements.

```
<http://dbtune.org/jamendo/artist/5> owl:sameAs
    <http://dbtune.org/musicbrainz/resource/artist/
0781a3f3-645c-45d1-a84f-76b4e4decf6d> .
<http://dbtune.org/jamendo/record/174> owl:sameAs
    <http://dbtune.org/musicbrainz/resource/record/
3042765f-67ba-49ef-ab28-45805fabef4a> .
<http://dbtune.org/jamendo/record/33> owl:sameAs
    <http://dbtune.org/musicbrainz/resource/record/
fade0242-e1f0-457b-99de-d9fe0c8cbd57> .
```

Having chosen this mapping we could go further, to also derive such statements for the tracks in these two albums.

#### 6.4.5.4 Evaluation

We now focus on a concrete interlinking which has been achieved using this graph-based interlinking algorithm, between two web datasets: Jamendo and Musicbrainz. We implemented our algorithm in SWI-Prolog [WHM08], as part of the `motools` open-source project. Our algorithm derived 10,944 equivalence statements for artist, record, and track resources so far. These statements allow user agents to merge detailed editorial information from Musicbrainz with the actual audio content, as well as tags, from the Jamendo dataset.

We evaluate the interlinking decisions made by the algorithm on artist resources. As we perform only one look-up on Musicbrainz, at the artist level, no tracks or records can be matched if the artist is not. In order to evaluate the quality of the interlinking, we take 60 random artists in the Jamendo dataset—1.7% of the artists available in this dataset. We manually check the correctness of the links derived by our graph-based interlinking algorithm for these artists, in the same way as in § 6.4.4. The results are shown in table 6.4.

Our algorithm made only two mistakes. These mistakes correspond to artists available through the look-up service, but not in the RDF dataset. The RDF dataset uses Musicbrainz database dumps, and is therefore not updated continuously. We also evaluate the same dataset with the constrained literal-based interlinking algorithm described in § 6.4.4. This algorithm got 33% of its interlinking decisions wrong for the artist testset, compared to 3% with our graph-based interlinking algorithm.

### 6.4.6 Summary and discussion

In order to automatically interlink structured web datasets, we investigated three algorithms. The first two algorithms (literal-based and constrained literal-based) are directly based on work from the database community on record linkage. We modelled the descriptions of our web resources as flat records, and we tried to match these records from one dataset to another. In our first algorithm, this record just holds the literals that are attached to the web resource. In the second algorithm, it also includes information about the type of the resource. We found that these two first algorithms were not reliable, and could only work correctly on specific datasets. We designed a new algorithm in § 6.4.5, exploiting the graph structure of web information. This algorithm continuously explores new related data until it can take an interlinking decision. We found that this algorithm performed well, relatively to the criteria in § 6.4.2.1—it makes a low number of incorrect positive interlinking decisions, and it does not require any manual post-processing. The links derived by this algorithm are now accessible as part of the Jamendo dataset, as illustrated in our example in § 6.3.2.

A possible extension of our graph-based interlinking algorithm is to associate RDF properties with weights, in order to start from the most informative property. For example, we might prefer to follow a `foaf:made` link first, when having the choice between this property and `mo:genre`. Another possible extension of this algorithm is to perform constrained literal lookups in $B$ at each step, therefore providing new possible graph mappings each time. This helps ensure we still find the correct mapping in the case that our initial literal lookup does not include the correct resource among its results. In an artist interlinking scenario, the correct target artist might be listed as having a different name in $B$ as in $A$, such that it does not feature in the results of our initial literal lookup. However, we might have some clues about who this artist is from the names of the albums they produced, and so performing additional literal lookups (on the album titles) may allow us to find the correct artist and hence the correct mapping. Such a variant of our algorithm is implemented in the GNAT software described in § 8.1.1.

Our algorithm works on the assumption that the two datasets to interlink conform to the same ontology, or that there is a one-to-one mapping between terms in the first ontology and terms in the second. In the latter case, we must consider performing an ontology matching task [ES07], and include the resulting correspondences between ontology terms in our algorithm. We could also combine the two tasks using an approach similar to [MNJ05], modelling the different correspondences at the ontology or at the instance level in a Bayesian network.

It is still possible that, even when embedding an ontology matching task in our algorithm, the relationships between matched resource is not well captured by an `owl:sameAs` link. For example, a dataset might consider London as a political entity, whereas another dataset might consider London as an area. A political entity is not described in the same way as a geographic area, so our algorithm would probably derive a positive non-link decision. However, these two resources are obviously related. We may need to introduce a *grey zone* in the decision step of our algorithm, as depicted in Figure 6.6. Inside this zone, we would consider using a co-reference service such as the Consistent Reference Service [GMJ$^+$07] to assert a relationship with weaker semantics than `owl:sameAs` between the two resources. Above this zone, we would draw a positive link decision. Under this zone, we would draw a positive non-link decision.

Also, our graph-based interlinking algorithm is designed for the case where a meaningful similarity measure between pairs of individual resources is available (here, using string similarity

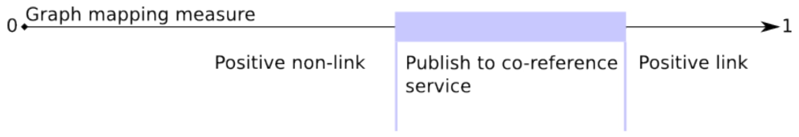


Figure 6.6: Modifying the decision step of our graph-based interlinking algorithm to handle weak associations between resources using a co-reference service

of labels attached to them with certain predicates, but we could also consider other similarity measures for other types of resources, e.g. a content-based audio similarity). In a linking scenario where there is no particularly good similarity measure on individual resources and the graph structure is therefore the most salient factor for correct linking, another algorithm may be more appropriate. In this case, Melnik's "Similarity Flooding" [MGMR02] approach could be used, which prioritises graph structure rather than node similarity, relying on a post-processing stage to filter out unsuitable mappings. However, for the example datasets depicted in Figure 6.5, Similarity Flooding succeeds to infer the mapping between the corresponding artist resources but fails to derive the mapping between corresponding records—all possible mappings are considered equally likely. Also, Similarity Flooding is difficult to adapt to an online context, where we discover parts of the graph as we go.

## 6.5 Quantifying structured web data

In this section, we try to identify different global characteristics of the datasets depicted in Figure 6.2. This will allow us to put our contribution on structured music-related data, described in the two previous sections, in a wider context. Although we do not pretend to give a metric quantifying whether a dataset is "good" or "bad", we want to derive some metrics to evaluate some of its key characteristics.

We focus on studying the Linking Open Data datasets [HHRH08] described in § 6.1, as they hold large amounts of publicly accessible real-world web data. These datasets cover a wide range of domains, and are either centralised (i.e. a single entry-point to the whole dataset is provided) or decentralised (i.e. no single entry-point is available, small pieces of structured data are provided by a range of different parties).

We define two measures. The first measure captures the relative size of the different datasets. The second measure characterises the amount of linkage to further datasets published by a particular dataset.

### 6.5.1 Metrics

Finin and Ding [FD06] argue that a single metric taking into account only the number of public RDF documents on the Web is an overly simple measure of structured web data. Indeed, it does not capture the actual web aspect—how interlinked are these RDF documents? This measure is also hard to estimate, as the biggest search engines do not index all available RDF documents. Ding and Finin [DF06] estimate the number of available RDF documents on the order of $10^7$ to $10^9$, in May 2006. We do not consider estimating such a global size, as we want to derive metrics characterising datasets, and see how the datasets compare to each other according to those.

We consider the number of facts (RDF statements) held by a particular dataset as a first metric $\mu$.

$$\mu = |D| \tag{6.3}$$

where $D$ is the set of RDF statements $(s, p, o)$ in the dataset.

We also quantify how isolated a particular dataset is or how much it links to other datasets. We therefore consider another metric $\nu$, defined as follows.

$$\nu = \frac{|E| \cdot 100}{\mu} \tag{6.4}$$

where $E$ is the set of RDF statements $(s, p, o)$ in the dataset such that $s$ or $o$ are web identifiers in other datasets

### 6.5.2 Results

Table 6.5 gives an overview of these two metrics for datasets within the Linking Open Data project. For some datasets, the values are approximates. They correspond to datasets in which the generation of an RDF representation is done on-demand (like the ones using the software described in § 6.2.1 and § 6.2.2) and for which there is no easy way of evaluating their actual size. In the case of the chord service, the representation (including a link towards a remote chord depiction service) is generated from the form of the web identifier. Therefore, there is an unlimited number of RDF statements available through that service.

In the case of datasets with no central point of access, these metrics are also unavailable. Such decentralised datasets are in fact defined by a particular web ontology. The criteria for a document to be part of such a dataset is indeed the fact that it quotes some of the terms defined in such an ontology. These datasets include SIOC, interlinking online communities [BBF08], and FOAF, describing persons and connections between them [BM07]. Ding et al. [DZFJ05] analyse some aspects of FOAF information available on the Web. They sample some FOAF information using a web crawler, and analyse the resulting dataset. In particular, they show that a small number of hosts serve most of the data. They also perform an analysis of the social network specified by this FOAF information.

We could also consider the Music Ontology as being such a decentralised dataset, and consider not only the big Music Ontology information publishers, but also smaller documents mentioning Music Ontology terms. For example, the first six datasets in table 6.5 all use Music Ontology terms. Available hAudio Microformats [Spo08] can also be included in such a dataset, as a GRDDL transformation (as mentioned in § 2.2.4.4) allows them to be translated to Music Ontology RDF. Also, a number of FOAF profiles include some information about the musical activities of a particular person (performances, musical interests, etc.) that use some Music Ontology terms.

### 6.5.3 Towards better metrics for interlinked web data

Our measures $\mu$ (size) and $\nu$ (interlinkage) for interlinked web datasets are limited in terms of the characteristics they capture. Also, our interlinkage measure is difficult to use as a basis for comparison amongst web datasets. For example, a dataset linking to geographical locations

| **Dataset** | $\mu$ (millions) | $\nu$ (%) |
|---|---|---|
| Magnatune | 0.322 | 0.072 |
| Jamendo | 1.1 | 0.454 |
| BBC John Peel | 0.277 | 0.758 |
| AudioScrobbler | $\approx$ 600 | N/A |
| Musicbrainz | $\approx$ 60 | $\approx$ 0.11 |
| MySpace | $\approx$ 12,500 | 0 |
| Chord service | N/A | N/A |
| Surge radio | 0.218 | N/A |
| DBLP | 28 | 0 |
| DBpedia | 109.75 | 2.4 |
| Riese | $\approx$ 3,000 | $\approx$ 0 |
| FOAF | N/A | N/A |
| SIOC | N/A | N/A |
| Geonames | 93.9 | 0.092 |
| GovTrack | 1,012 | 0.002 |
| lingvoj | 0.01 | 10 |
| Ontoworld | 0.06 | 0.167 |
| OpenCyc | 0.25 | 0 |
| Open-Guides | 0.01 | 0 |
| Project Gutenberg | 0.01 | 0 |
| Revyu | 0.02 | 3 |
| SemWebCentral | 0.01 | 0 |
| SW Conference Corpus | 0.01 | 5 |
| W3C Wordnet | 0.71 | 0 |
| World Factbook | 0.04 | 0 |
| Total | $\approx$ 17,407 | 0.021 |

Table 6.5: $\mu$ and $\nu$ metrics for datasets within the Linking Open Data project, as of May 2008

within Geonames through the use of `foaf:based_near` is likely to have a higher $\nu$ value that the same dataset linking to Geonames through `owl:sameAs`. To illustrate that problem, the following dataset would have a $\nu$ value of 100%.

```
:item1 foaf:based_near geo:location1.
:item2 foaf:based_near geo:location1.
```

The following dataset, equivalent to the above dataset, would have a $\nu$ value of 33%.

```
:item1 foaf:based_near :location1.
:item2 foaf:based_near :location1.

:location1 owl:sameAs geo:location1.
```

We therefore need a more accurate measure for interlinkage. For example, we propose to always take the lowest $\nu$ value, obtained by transforming the dataset as illustrated in the two above examples. This results in only considering `owl:sameAs` interlinkage.

Other interesting metrics would also quantify the *quality* of the actual data. For example, a more complete evaluation would use the metrics we define in table 6.6. It remains future work to do an extensive evaluation of currently available web datasets using these metrics. It would also be interesting to see how all these different metrics evolve over time.

| Feature captured | Example metric |
|---|---|
| Utility | Number of web resources linking to resources within the dataset |
| Interlinking accuracy | Mean accuracy of the interlinking algorithm used (see § 6.4.2) |
| Accuracy | Comparison between original data source and web dataset |
| Currency | Update cycle with regards to the update cycle of the original dataset |
| Stability | Number of web identifiers that changed in a fixed period of time |
| Reliability | For how long the web dataset has been made available |
| Licensing | Amount of statements with a clear license |

Table 6.6: Further characteristics to capture in an evaluation of web datasets, along with example metrics

### 6.5.4 Scaling the tools

The amount of structured web data published within the scope of the Linking Open Data project is bigger than what any of the current RDF stores can handle. It is difficult, given the state of the tools, to create a massive aggregation of *all* this structured web data and to provide a querying facility on top of it. However, we still want to perform cross-dataset queries such as "give me the list of records made by artists belonging to that political movement", involving information from Musicbrainz and DBpedia. In order to perform such queries, we consider the following alternatives.

- A first solution is to create an aggregation specifically designed to answer a particular query. For example, the Semantic Web Client Library or our SWIC software mentioned in § 2.2.5 aggregate structured web data starting from the web identifiers appearing in an input query. For example, we consider the following query ("give me poets born in the same region in which the band who made a particular record is based"):

  ```
  SELECT ?poet
  WHERE {
  <http://dbtune.org/jamendo/record/33>
    a mo:Record;
    foaf:maker ?artist.
  ?artist
    foaf:based_near ?place.
  ?city
    p:department ?place.
  ?poet
    a yago:Poet110444194;
    p:birthPlace ?city }
  ```

  Processing this query with the Semantic Web Client Library or SWIC leads to accessing the following web identifiers, in DBTune, Geonames, and DBpedia:

  1. `http://dbtune.org/jamendo/record/33`
  2. `http://dbtune.org/jamendo/artist/5`
  3. `http://sws.geonames.org/2991627/`
  4. `http://dbpedia.org/resource/Moselle`

5. `http://dbpedia.org/resource/Metz`
6. `http://dbpedia.org/resource/Paul_Verlaine`

However, this solution is slow, due to the number of web identifiers to access at querying time.

- Another solution is to create a *focused aggregation*, aiming at answering a particular type of query. For example, the BBC Playcount end-point described in § 6.3.6 holds aggregated information about the involved artists and BBC brands. Therefore, the end-point can answer quickly queries similar to the following query ("give me BBC programmes in which the Beatles were featured at least ten times"):

```
SELECT ?brand ?title ?count
WHERE {
?artist
  a mo:MusicArtist ;
  foaf:name "The Beatles" .
?pc
  pc:object ?artist ;
  pc:count ?count .
?brand
  a po:Brand ;
  pc:playcount ?pc ;
  dc:title ?title .
FILTER (?count>10) }
```

  We describe in Appendix B our approach to describe such aggregations, in order to automatically redirect Semantic Web queries towards relevant aggregations.

- If all the datasets we want to query already provide SPARQL end-points, we might not want to go through the burden of creating a focused aggregation involving some of their data. Instead, we may want to issue *federated* queries [LWB08, Ver08] — queries that involve multiple end-points.

## 6.6 Summary

In this chapter, we described a number of tools allowing us to easily publish structured data on the Web. We mentioned a project gathering such structured data across a number of domains, from bio-informatics to music-related or social networking information. We described a number of music-related datasets we published within this project. We specified an effective algorithm automatically drawing links between such datasets. Finally, we measured the size and the interlinkage of these different datasets.

The variety of the published music-related datasets illustrates the expressiveness of the Music Ontology described in chapter 3 and the flexibility of Semantic Web ontologies. We now have an integrated information environment gathering editorial information, listening habits, musicological information, social networks, Creative Commons content, encyclopedic information, and

European statistics. We have a large a large web of data that can be used by automated agents for a wide range of purposes, as illustrated in chapters 7 and 8.

# Chapter 7

# Automated music processing agents

We described in § 5.2 a knowledge representation framework (N3-Tr) allowing us to publish and interlink music processing workflows on the Web. The structured web data described in chapter 6 and N3-Tr workflows can be aggregated and interpreted by automated agents, which can use the publication mechanisms described in § 5.3 to publish the resulting information on the Web. The resulting information can then be used by further automated agents deriving new information, or by other user agents for e.g. music collection management purposes, as we will see in § 8.1.

We describe in § 7.1 our implementation of such a N3-Tr agent. We detail in § 7.2 the available interfaces for accessing the information this interpreter is able to derive. We give in § 7.3 some examples of music-related information that our N3-Tr agent is making available on the Web.

## 7.1 Implementation of a N3-Tr agent

We detail here an implementation[1] of an automated agent able to aggregate and interpret formulæ expressed in our N3-Tr knowledge representation framework, along with other information available on the Web. This agent provides on-demand access to the results it is able to derive. Our implementation relies on SWI-Prolog and its Semantic Web library [WHM08].

### 7.1.1 Overview

A component interpreting N3-Tr formulæ is not fundamentally different from any Semantic Web user agent, such as the user agents mentioned in § 2.2.5. The only difference lies in the reasoning that can be performed on top of the aggregated information. A fully-fledged N3-Tr agent can apply N3-Tr workflows to derive and publish more information.

Our implementation is built on top of a persistent database, storing information gleaned from RDF and N3-Tr documents on the Web as well as cached computations. The state of the RDF information held within this store corresponds to the state **D** defined in § 5.2.3.2.

A component interprets the N3-Tr formulæ available in this database. A web access module allows the agent to retrieve and aggregate web information. A module handles the publication of the results the component is able to derive. This architecture can be depicted as in Figure 7.1.

[1] The source code is available at `http://code.google.com/p/km-rdf/`.

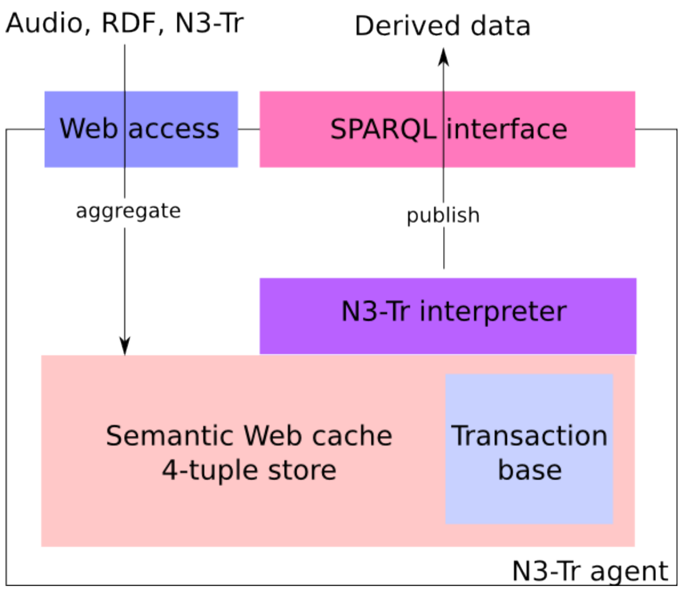


Figure 7.1: Architecture of our N3-Tr agent

### 7.1.2 Parsing and storing N3

In order to aggregate RDF and N3-Tr information, we need to have RDF and N3 parsers. We use the RDF parser from the SWI-Prolog Semantic Web library, supporting the RDF/XML and Turtle syntaxes described in § 2.2.4.3.

The N3 parser was built on top of the SWI-Prolog Turtle parser[2] and produces a set of quads (4-tuples) of the following form:

$$(\mathsf{subject}, \mathsf{predicate}, \mathsf{object}, \mathsf{context}) \tag{7.1}$$

Where $\mathsf{context}$ identifies the graph literal[3] in which the RDF triple $(\mathsf{subject}, \mathsf{predicate}, \mathsf{object})$ is defined. In case the triple is not held within a particular graph literal, we use this fourth argument to store the location of the N3-Tr document from which the triple was gleaned. The $\mathsf{context}$ identifier can be used in other RDF triples.

Existentially quantified variables are parsed as RDF blank nodes, as handled by the SWI-Prolog Semantic Web library i.e. Prolog atoms starting with `__bnode`. Universally quantified variables are also represented internally as Prolog atoms, starting with `__uqvar`. Graph literals are identified by Prolog atoms starting with `__graph`.

N3 *paths* (e.g. `:performance!mo:performer!foaf:name`, meaning "the name of the performer involved in the performance `:performance`") are expanded into a set of RDF triples involving existentially quantified variables, as specified in [BL06b]. We also expand the different keywords defined in N3, such as the keyword `=>` for the web identifier `log:implies`, or the keyword `a` for `rdf:type`.

**Example 18.** For example, the following N3 code available in the document `document.n3`, illustrating a part of the definition of a performer drawn in § 3.3.2.2, would be parsed as in table

[2]The corresponding Definite Clause Grammar is available at `http://code.google.com/p/km-rdf/source/browse/trunk/n3/n3_dcg.pl`

[3]We here move away from graph literals to adopt explicitly named graphs. This is an implementation artifact allowing us to gather all of our information in a 4-tuple store.

| Subject | Predicate | Object | Context |
|---|---|---|---|
| `__uqvar_a` | `rdf:type` | `mo:Performer` | `__graph1` |
| `__bnode3` | `mo:performer` | `__uqvar_a` | `__graph2` |
| `__bnode3` | `rdf:type` | `mo:Performance` | `__graph2` |
| `__graph1` | `log:implies` | `__graph2` | `document.n3` |

Table 7.1: Parsing the N3 document `document.n3`. Node identifiers starting with `__uqvar` correspond to universally quantified variables, whereas node identifiers starting with `__bnode` correspond to existentially quantified variables and node identifiers starting with `__graph` correspond to graph literals.

7.1:

```
{?a a mo:Performer} => {_:p mo:performer ?a; a mo:Performance}.
```

Universally quantified variables, e.g. `?a` in this example, are mapped to the same Prolog atom, e.g. `__uqvar_a`, within a statement. If we were to re-use `?a` in another statement, we would map it to another Prolog atom.

Existentially quantified variables, e.g. `_:p` in this example, are mapped to the same Prolog atom, e.g. `__bnode3`, within a document. If we were to re-use `_:p` in the same document, we would map it to the same Prolog atom. If we were to re-use `_:p` in another document, we would map it to a different Prolog atom.

After parsing a N3 document, the resulting quads can be stored within the SWI-Prolog quad store. The SWI-Prolog RDF store indeed handles an extra element along with an RDF triple, usually used to describe the origin of the triple. The only change that needs to be made to the implementation of the SWI-Prolog quad store is to add a support for literals as the first element of an RDF triple.

### 7.1.3 N3-Tr interpreter

The database state **D** defined in § 5.2.3.1 corresponds to the state of the RDF information held within the SWI-Prolog quad store. Queries to that state are made through the `rdf` Prolog predicate of arity 3.

**Example 19.** The following goal will query the current state for a RDF triple $(\mathsf{a}, \mathsf{b}, \mathsf{c})$.

```
?- rdf(a,b,c).
```

The ins predicate defined within the transition oracle in § 5.2.3.1 is captured by the `cache` Prolog predicate of arity 3.

**Example 20.** The following goal will make the database evolve from a state **D** to a state $\mathbf{D} + \{(\mathsf{a}, \mathsf{b}, \mathsf{c})\}$.

```
?- cache(a,b,c).
```

We define four new Prolog operators, `>>`, `#`, `o` and ¬, respectively corresponding to the serial conjunction (the $\otimes$ connective in CTR, as defined in § 5.2.2), the concurrent conjunction (the $|$ connective), the isolation (the $\odot$ operator) and the negation (the ¬ operator). The `#` Prolog operator has a higher precedence than the `>>` operator: `(a >> b # c)` is equivalent to `((a >> b) # c)`. The `o` and ¬ Prolog operators have a lower precedence than both `#` and `>>`. We implemented an interpreter supporting these operators, based on two previous efforts of Prolog-based transaction logic interpreters [Hun96, Sle00].

**Example 21.** The following goal will make the database evolve from a state $\mathbf{D}$ to a state $\mathbf{D} + \{(a, b, c), (c, d, e)\}$ through an intermediate state $\mathbf{D} + \{(a, b, c)\}$ if $(e, f, g)$ is available in the final state.

```
?- cache(a,b,c) >> cache(c,d,e) >> rdf(e,f,g).
```

### 7.1.4 Built-in predicates

Our N3-Tr agent holds a number of signal processing built-in predicates, provided by the SWIAudio libraries[4]. These predicates correspond to $\mathbf{B}$, as defined in § 5.2.3.2. These built-ins are triggered when evaluating the corresponding `rdf(S,P,O)` predicates. In case the built-in needs several inputs or parameters, we use a Prolog list holding them. In case the built-in produces several outputs, we also use a Prolog list.

**Example 22.** A symmetrised Kullback-Leibler divergence between `model1` and `model2` (see § $A.4.3$) is triggered when evaluating the following predicate.

```
?- rdf([model1,model2],sig:jsdiv,Dim).
```

The resulting divergence is bound to the variable `Dim`.

We also integrated the Vamp feature extraction plugin architecture [Can] within our N3-Tr agent, as it allows us to easily create new built-in predicates for musical audio analysis.

### 7.1.5 Tabled predicates

As mentioned in § 5.2.3.2, a predicate can be declared as *tabled*. In that case, it may trigger state changes when evaluated, through the execution of the `table` transaction defined in § 5.2.3.2. Of course, the changes triggered by `table` depend on its determinism and on previous evaluations, as described in § 5.2.3.2. The N3-Tr interpreter currently handles the two following determinism categories.

- Deterministic (`owl:FunctionalProperty`) – Given a valid set of inputs, the predicate gives a unique set of outputs. If the predicate is tabled and has not yet been evaluated, the evaluated predicate is stored in the database. If the predicate is tabled and has already been evaluated, the previous evaluation is retrieved, without modifying the state of the database;

- Multi-solution (`ctr:MultiSolution`) – Given a valid set of inputs, the predicate gives one or more sets of outputs. If the predicate is tabled and previous evaluations are not yet in the store, all possible evaluations of the predicate are stored in the database. If the predicate is tabled and has already been evaluated, previous evaluations are retrieved, without modifying the state of the database.

All these behaviours are captured as `table` transactions in our implementation, similar to the `table` transaction defined in (5.13). For example, the deterministic predicate case is captured by the following transaction.

[4] see `http://code.google.com/p/km-rdf/`

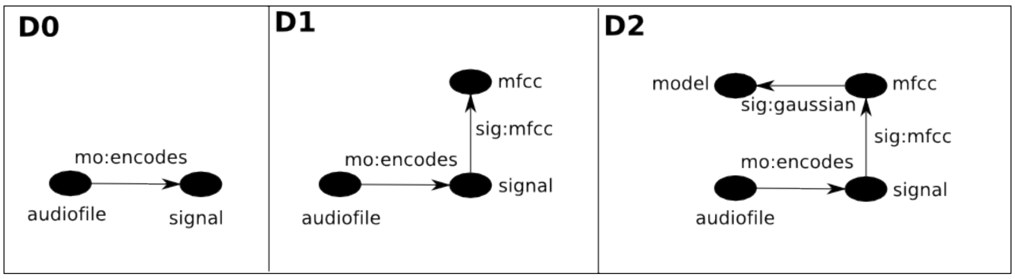


Figure 7.2: Depiction of the different database states involved by the transaction `table(signal,sig:mfcc,M) >> table(M,sig:gaussian,M)`. $D0$ is the initial state, $D1$ is an intermediate state, and $D2$ is the final state.

```
table(S,P,O) :-
   rdf(P,rdf:type,owl:‘FunctionalProperty’) >> rdf(S,P,O) >> cache(S,P,O).
```

We first check if the precondition is satisfied i.e. the predicate must be deterministic. Then, we evaluate the predicate and we store it.

**Example 23.** As in Example 7, the predicate `sig:mfcc` associates a signal to the corresponding MFCCs. The predicate `sig:gaussian` fits such coefficients in a Gaussian model. We consider the built-in predicates `sig:mfcc` and `sig:gaussian` as deterministic. Evaluating the following goal makes the database move through the states depicted in Figure 7.2.

```
?- table(signal,sig:mfcc,M) >> table(M,sig:gaussian,Model).
```

We first trigger the MFCC computation, bind the variable `M` to the MFCCs, and insert the corresponding ground RDF statement, linking the signal to these MFCCs, in the database. We then trigger the modelling of these MFCCs, and bind `Model` to the corresponding model. We update the database with the corresponding ground RDF statement, linking the MFCCs to their model.

### 7.1.6 Compiling N3-Tr rules

The stored N3 documents can hold N3-Tr rules, as described in § 5.2.3.3. Such rules are compiled by our N3-Tr agent to Prolog rules involving the Transaction Logic operators described in § 7.1.3 and the `table` predicate described in § 7.1.5.

We also use the *skolemisation* technique [VDO03, SOS$^+$08] to handle existentially quantified variables in the conclusion part of such rules. The skolemization process consists of replacing existentially quantified variables by a function of the universally quantified variables involved in the rule. The rule is then translated to its *skolem normal form*: it only involves universally quantified variables and is satisfiable if the original rule is. Also, rules that involve $N$ predicates in their conclusion part are translated to $N$ Prolog clauses. Therefore, our N3-Tr rules are translated to a set of concurrent Horn rules, as defined in § 5.2.3.4. These compiled Prolog clauses constitute our transaction base, operating over the RDF information available within our quad store.

We now give two examples of such compiled N3-Tr rules.

**Example 24.** The classical inference rule[5] mentioned in § 7.1.2 (“if someone is a performer, then he contributed to a performance”) would be compiled as follows.

[5]Concurrent Transaction Logic is a conservative extension of the classical predicate logic — it reduces to classical logic for formulæ that do not cause state transition.

```
rdf(individual('__skolem1',[Performer]),mo:performer,Performer) :-
        rdf(Performer,rdf:type,mo:'Performer').
rdf(individual('__skolem1',[Performer]),rdf:type,mo:'Performance') :-
        rdf(Performer,rdf:type,mo:'Performer').
```

In this example, we see that the rule is translated into its skolem normal form. The Prolog term `individual('__skolem1',[Performer])` corresponds to the existentially quantified performance. Also, the rule translates to two Prolog clauses, as there are two predicates involved in its conclusion part.

**Example 25.** The content-based audio similarity N3-Tr rule in § 5.2.4.3 would be compiled as follows.

```
rdf([Signal1,Signal2],sim:div,Div) :-
        (table(Signal1,sig:mfcc,MFCC1) >> table(MFCC1,sig:gaussian,Model1) #
        table(Signal2,sig:mfcc,MFCC2) >> table(MFCC2,sig:gaussian,Model2)) >>
        table([Model1,Model2],sig:jsdiv,Div).
```

To compute a divergence between two audio signals, we compute concurrently a timbre model for each of them and then we compute a divergence between these two models.

### 7.1.7 Network access

We now have a functional N3-Tr interpreter, working on top of a local quad store holding RDF information and N3-Tr rules. However, we still need to take advantage of the fact that we are using web identifiers throughout our representation framework, as highlighted in § 5.2.5.

Prior to the evaluation process, the N3-Tr agent accesses the mentioned web identifiers and aggregates or updates their representation in its quad store. This enables the discovery of new workflows, new RDF information or new audio files at evaluation time.

**Example 26.** When evaluating the following goal, the N3-Tr agent accesses `sim:div`, therefore giving access to the content-based audio similarity workflow specified in § 5.2.4.3.

```
?- rdf([signal1,signal2],sim:div,Div).
```

**Example 27.** When evaluating the following goal, the N3-Tr agent accesses `sig:mfcc`.

```
?- rdf(signal,sig:mfcc,M).
```

This identifier gives access to a workflow detailing how to get from a signal to MFCCs. This might be useful in case the agent evaluating the query does not have any built-in predicates to derive MFCCs, but does know how to derive lower-level results (Fourier transform, Mel filter banks, logarithm, discrete cosine transform). The N3-Tr agent then acquires at evaluation time a worklow joining up these different processes.

**Example 28.** When evaluating the following goal, the N3-Tr agent accesses
`http://moustaki.org/foaf.rdf`.

```
?- rdf('http://moustaki.org/foaf.rdf#moustaki',foaf:depiction,Img).
```

This identifier gives access to more information about that person, including depictions.

### 7.1.8 Future work on the N3-Tr agent

It remains future work to handle inference over OWL-DL constructs that are within the scope of Description Logics Programs [GHVD03] — axioms that are within the Description Horn Logic fragment of the $\mathcal{SHOIN}(D)$ description logic, as discussed in § 5.4.4.

Future work also includes the development of a graphical environment for creating, editing and publishing N3-Tr rules. Taverna [OGA+06] would provide a good basis for a stand-alone application, and "Semantic Web Pipes" [MPT07] would provide a good basis for on-line editing of N3-Tr rules.

## 7.2 Publication of derived information

So far, we are only able to issue N3-Tr goals to the Prolog interpreter. The content-based audio similarity defined in § 5.2.4.3 can be accessed by issuing the following at the Prolog prompt:

```
?- rdf([:signal1,:signal2],sim:div,Div).
```

We want such derivable information to be accessible publicly on the Web. We therefore design two interfaces for accessing such information, implementing the mechanisms described in § 5.3.

### 7.2.1 SPARQL interface

Our N3-Tr agent includes a SPARQL interface. This interface can be used to remotely issue N3-Tr goals, using the SPARQL to N3-Tr mapping described in § 5.3.1.1. For example, the following SPARQL query triggers the N3-Tr rule defined in § 5.2.4.3 for two audio signals and binds the resulting divergence to the variable `?div`.

```
SELECT ?div ?duration
WHERE {
(:signal1 :signal2) sim:div ?div }
```

### 7.2.2 Web interface

The web publication interface specified in § 5.3.2 is easily set up by using our UriSpace software described in § 6.2.3. UriSpace allows us to create web identifiers for RDF documents corresponding to the result of particular CONSTRUCT or DESCRIBE queries on the SPARQL end-point exposed by the N3-Tr agent. It remains future work to automatically derive such web identifiers, by analysing the workflows and the information known by the N3-Tr agent.

Coupled with the network access mechanism described in § 7.1.7, this web interface allows us to distribute computation over several instances of our N3-Tr agent, as in the scenario described in § 5.3.3.

## 7.3 N3-Tr agent on DBTune

We made an instance of our N3-Tr agent[6] available within our DBTune service described in § 6.3. This agent gives on-demand access to all the information it can derive by applying the N3-Tr

[6]see `http://dbtune.org/henry/`

workflows it holds to the web information it aggregates. This agent is pre-loaded with several N3-Tr workflows. We here describe two of them.

### 7.3.1 Keys

The first N3-Tr workflow describes how to derive events corresponding to changes of musical keys[7] from an audio file available on the Web. The derived events are described in terms of our Music Ontology framework.

```
:key_translation =
  ((1.0 key:Cmajor) (2.0 key:Csmajor) (3.0 key:Dmajor) (4.0 key:Dsmajor)
  (5.0 key:Emajor) (6.0 key:Fmajor) (7.0 key:Fsmajor) (8.0 key:Gmajor)
  (9.0 key:Gsmajor) (10.0 key:Amajor) (11.0 key:Asmajor) (12.0 key:Bmajor)
  (13.0 key:Cminor) (14.0 key:Csminor) (15.0 key:Dminor) (16.0 key:Dsminor)
  (17.0 key:Eminor) (18.0 key:Fminor) (19.0 key:Fsminor) (20.0 key:Gminor)
  (21.0 key:Gsminor) (22.0 key:Aminor) (23.0 key:Asminor) (24.0 key:Bminor)) .

sig:keys a ctr:TabledPredicate .
mo:encodes a ctr:TabledPredicate .

{
   (qvp:qm-vamp-plugins "qm-keydetector" ?signal ("key")) vamp:transform ?keys
} => {
      ?signal sig:keys ?keys
} .

{
  ?af mo:encodes ?signal .
  ?signal mo:time [tl:timeline ?stl] .
  ?signal sig:keys ?keys .
  ("key" ?start _:fduration ?keys) list:in ?keys .
  ?key_number list:in ?keys .
  (?key_number ?key) list:in :key_translation .
} => {
_:tonalchange a af:KeyChange ;
    rdfs:label "Tonal region delimited by the key" ;
    event:time [ tl:timeline ?stl ; tl:at ?start ] ;
    to:new_key ?key } .
```

The first step is to cache and decode the audio file. These two steps are wrapped within an N3-Tr rule at `mo:encodes`. Once we have the corresponding signal in store, we use a Vamp plugin extracting key changes by matching the results of the chromagram transformation described in § *A*.3 to key profiles [NS07]. This plugin gives us a list of 4-tuples including information about the timing of a key change and the new key. Of course, other workflows could be substituted at

[7]The key is defined in [Sad80, Vol. 10] as the quality of a musical composition or passage that causes it to be sensed as gravitating towards a particular note.

| time | key |
|---|---|
| 0.0 | to:key/Cmajor |
| 2.97209 | to:key/Cminor |
| 62.4134 | to:key/Bmajor |
| 67.6145 | to:key/Fsmajor |
| 69.1005 | to:key/Bmajor |
| 88.4189 | to:key/Cminor |
| 124.827 | to:key/Fminor |
| 130.771 | to:key/Cminor |
| 132.257 | to:key/Bmajor |
| 138.944 | to:key/Emajor |
| 151.575 | to:key/Gsmajor |
| 155.29 | to:key/Bmajor |
| 156.033 | to:key/Cminor |
| 185.011 | to:key/Fminor |
| 192.441 | to:key/Bmajor |
| 216.218 | to:key/Cminor |
| 227.363 | to:key/Csminor |
| 291.262 | to:key/Fsminor |

Table 7.2: Results of the SPARQL query in § 7.3.1, giving key changes derived from the audio file `http://dbtune.org/audio/Both-Axel.ogg`

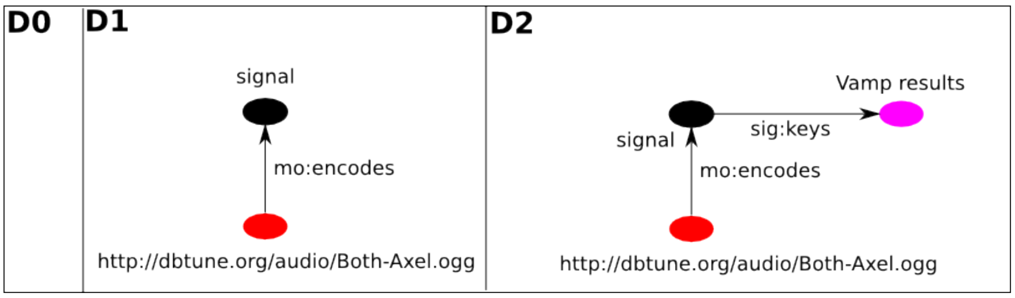

D0
D1
signal
mo:encodes
http://dbtune.org/audio/Both-Axel.ogg
D2
Vamp results
sig:keys
signal
mo:encodes
http://dbtune.org/audio/Both-Axel.ogg

Figure 7.3: Depiction of the different database states involved in evaluating the SPARQL query in § 7.3.1. We start at the empty state **D0**. We cache and decode the newly discovered audio file, which leads us to the state **D1**. Then, we apply the key extraction Vamp plugin, which leads us to the state **D2**. The red node corresponds to the audio file. The violet node corresponds to the derived Vamp results.

this stage, for example specifying these different steps. We give meaning to the results derived by this plugin by mapping them to our ontological framework described in chapter 3. These results correspond to key change events on the timeline of the corresponding audio signal.

For example, we can now issue the following SPARQL query:

```
SELECT ?time ?key WHERE {
<http://dbtune.org/audio/Both-Axel.ogg> mo:encodes ?sig.
?sig mo:time ?t.
?t tl:timeline ?tl.
_:evt a af:KeyChange;
        event:time [tl:at ?time; tl:timeline ?tl] ;
        af:new_key ?key }
```

The different database states involved in the evaluation of that query in an empty state are depicted in Figure 7.3. The results of such a query are available in table 7.2.

### 7.3.2 Content-based similarity

The second N3-Tr workflow pre-loaded within our DBTune instance is the content-based audio similarity workflow described in § 5.2.4.3, except that it starts with audio files, not audio signals — it therefore extends the workflow with a decoding step. For example, we can issue the following SPARQL query in the **D3** state depicted in Figure 7.3.

```
PREFIX sim: <http://purl.org/ontology/similarity/>
SELECT ?div
WHERE {
(<http://dbtune.org/audio/Den-Nostalia.ogg>
 <http://dbtune.org/audio/Both-Axel.ogg>) sim:div ?div }
```

The different database states involved in the evaluation of that query are depicted in Figure 7.4.

The thresholding rule in § 5.2.4.3 is applied by our SBSimilarity service. The database of this second agent currently holds distance information for around 300,000 tracks. Web identifiers like those described in § 5.3.2.1 were set up using our UriSpace software described in § 6.2.3. This provides on-demand computation of between-track similarities, and track resources are linked to the corresponding Musicbrainz resources described in § 6.3.5, along with audio previews and purchase pages on the Amazon music store. We describe an application using this service in § 8.3.2.

### 7.3.3 Others

Using the mechanism described in § 7.1.7, new workflows can be registered easily, by just mentioning a web identifier giving access to them within a particular query. As highlighted in § 7.1.7, in the case where we start an agent with an empty transaction base, the content-based similarity workflow can be loaded within the agent by just mentioning the `sim:div` web identifier in a query, such as in the query in § 7.3.2. When going through the initial aggregation step, the agent will update its transaction base with the N3-Tr workflow defined in § 5.2.4.3 and will be able to use this workflow thereafter.

Moreover, as the agent includes a Vamp plugin interface for audio analysis, new built-in predicates can be discovered by following links towards Vamp implementations. For example, the representation of `sig:chromagram` can link to the Vamp implementation of a chromagram computation, which can be used by the agent.

Another interesting built-in predicate is the `gnat:match` predicate, identifying a variant of the algorithm in § 6.4.5 described in § 8.1.1, which finds the corresponding web identifier in the Musicbrainz dataset described in § 6.3.5 for a particular audio file. Then, the following N3-Tr rule can be used to link to Musicbrainz all the results derived by our N3-Tr agent and involving an audio file.

```
{ _:track
    owl:sameAs ?musicbrainz_track;
    mo:available_as ?audiofile
 } <= {
   ?audiofile gnat:match ?musicbrainz_track } .
```

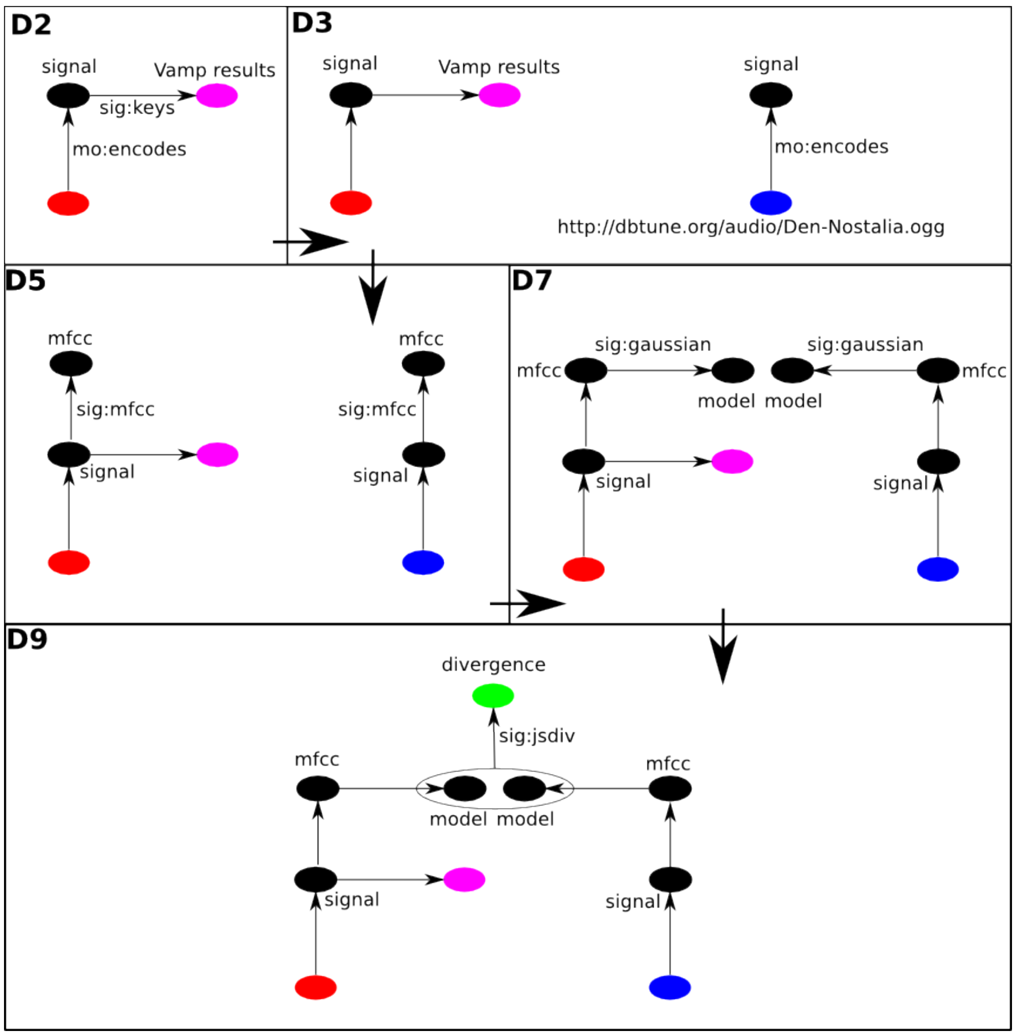


Figure 7.4: Depiction of the different database states involved in evaluating the SPARQL query in § 7.3.2. We start at the state **D2** depicted in Figure 7.3. We start by caching and decoding the previously unknown audio file, therefore going to the state **D3**. We then evaluate MFCCs concurrently, and end up in the state **D5**. We model these MFCCs, and end up in the state **D7**. We reach the final state **D9** after computing the divergence between the two models. The blue node corresponds to the newly discovered audio file. The green node corresponds to the derived divergence.

## 7.4 Summary

In this chapter, we described our implementation of a N3-Tr agent, interpreting formulæ expressed in our N3-Tr representation framework specified in chapter 5. This agent is able to aggregate N3-Tr music processing workflows available on the web as well as other structured web data, e.g. the datasets mentioned in chapter 6. It provides on-demand access to the results it is able to derive through two interfaces, SPARQL-based and web-based.

We described an available instance of our N3-Tr agent, along with two workflows that have been pre-loaded. The first workflow deals with the extraction of key change events, and the second workflow deals with a content-based divergence measure. New N3-Tr workflows and new structured web data can be discovered at querying time. This N3-Tr agent therefore implements a music-related 'Semantic Web agent' [BLHL01], processing machine-readable web information and musical objects to publish new web information.

Combined with the datasets available on the Semantic Web described in chapter 6, we are getting towards a rich and dynamic music-related web of data, encompassing a wide range of information and automatically enriched. We can start using this information environment for a wide range of applications, such as collection management, visualisation, search and recommendation.

# Chapter 8

# Case studies

In chapters 6 and 7, we described how the technologies developed in part II can be used to gather in a single environment a wide range of music-related information, from complex editorial metadata to information derived by automated agents. Having access to such a rich web of structured data allows us to design innovative applications, tackling a wide range of domains — from personal collection management to music recommendations. This chapter is focused around examples of such applications.

Our GNAT and GNARQL software creates an aggregation of structured web data describing a personal music collection (or any collection of audio items). GNARQL also includes a user interface, allowing a user to browse and find items in his collection in new ways. We describe these applications in § 8.1.

Syndication of web content (a section of a web site made available for other sites or applications to use) has become a common practise. In particular, syndication of audio content (*podcasts*) has become popular over the last years. However, it is extremely difficult to find podcasts, particular podcast entries, or segments of such entries. We describe our N3-Tr-based framework for tackling these issues in § 8.2.

Interlinking user profiles, social networks, content-based similarity statements, and relationships between artists allows us to consider new approaches for music recommendation. We present in § 8.3 our first steps at designing a music recommender system using the interlinked data described in chapters 6 and 7.

## 8.1 Personal collection management

Personal music collections can be part of the information environment described in chapters 6 and 7. As mentioned in § 3.3.1, our Music Ontology framework makes the same distinction as in FRBR between *manifestations* (all physical objects that bear the same characteristics, e.g. a particular album) and *items* (a concrete entity, e.g. my copy of the album on CD). A manifestation and the corresponding item are linked through the Music Ontology property `mo:available_as`, subsuming the FRBR *exemplar* property.

We detail a tool automatically deriving such links in § 8.1.1. Then, we detail in § 8.1.2 a tool aggregating structured web data from such links to create a database describing a personal music collection. Finally, we detail a user interface working on top of such an aggregation in § 8.1.3.

### 8.1.1 GNAT

Given a set of audio files in a personal audio collection, it is possible to keep track of the set of statements linking this collection to identifiers denoting the corresponding manifestations available elsewhere in the Web. These statements provide a set of entry points to the interlinked data described in the previous chapters, allowing one to access information such as the birth date of the artists responsible for some of the items in the collection, geographical locations of the recordings.

To this end, we have developed GNAT[1], which associates a personal audio collection to the Musicbrainz dataset. GNAT implements a variant of the automated interlinking algorithm described in § 6.4.5. GNAT first derives an RDF graph designed using the Music Ontology framework described in chapter 3. This graph describes the personal audio collection, using to this end the available metadata e.g. ID3 tags and the directory structure. Then, it applies the algorithm described in § 6.4.5 to interlink this graph and the Musicbrainz dataset described in § 6.3.5. GNAT implements a variant to the pseudocode in Appendix C, as it includes a constrained literal lookup at each step, therefore providing new possible graph mappings each time. This ensures our algorithm finds the correct mapping even if the initial literal (in the case of GNAT, the name of the artist) does not give access to the right resource in Musicbrainz. Indeed, we might have some clues about who this artist is from the names of the albums they made, and so performing additional literal lookups (on the album titles) may allow us to find the correct artist and hence the correct mapping.

GNAT outputs a set of RDF statements, linking local audio files and the remote manifestation identifiers.

```
<http://dbtune.org/musicbrainz/resource/track/80ee2332-9687-4627-88ad-
40f789ab7f8a> mo:available_as <file:///Music/go-down.mp3> .
<http://dbtune.org/musicbrainz/resource/track/b390e4d2-788c-4e82-bcf9-
6b99813f59e6> mo:available_as <file:///Music/dog-eat-dog.mp3> .
```

### 8.1.2 GNARQL

Then, we can aggregate information about a particular audio collection using as an input such RDF statements, providing entry points into the large web of structured data described in the previous chapters. A collection management tool then acts as an *aggregator* — it automatically creates a tailored database describing the managed collection.

Our GNARQL tool[2] explores some of these possibilities. GNARQL loads the links outputted by GNAT and crawls the Web for more information about the user's audio collection. GNARQL therefore provides a functionality similar to the Exposé tool [LH97] but in a linked data context. GNARQL creates an aggregation of structured web data focused on the user's audio collection. This aggregation is composed of information that is gathered from a completely distributed information environment. GNARQL exposes this aggregation through a SPARQL end-point. With the datasets currently available and interlinked, GNARQL is able to answer the following SPARQL queries.

[1]available as part of the motools project `http://www.sourceforge.net/projects/motools/`
[2]available as part of the same motools project

**Example 29.** "Give me tracks in my audio collection corresponding to performances in the Royal Albert Hall of works by German composers, written between 1800 and 1850."

```
SELECT ?track
WHERE {
?performance
  a mo:Performance ;
  event:place <http://dbpedia.org/resource/Royal_Albert_Hall> ;
  mo:recorded_as ?signal ;
  event:factor ?work .
?work a mo:MusicalWork .
?composition
  a mo:Composition ;
  event:product ?work ;
  event:agent ?composer .
  event:time [ tl:at ?date ] .
?composer p:born dbpedia:Germany .
?signal mo:published_as ?track .
FILTER(?date > "1800-01-01"^^xsd:date &&
  ?date < "1850-01-01"^^xsd:date) }
```

**Example 30.** "Give me audio files in my collection made by people who are engaged to each other."

```
SELECT ?audiofile1 ?audiofile2
WHERE {
?person1 rel:engagedTo ?person2 .
?track1 foaf:maker ?person1 ; mo:available_as ?audiofile1 .
?track2 foaf:maker ?person2 ; mo:available_as ?audiofile2 }
```

**Example 31.** "Give me artists in my collection, who made at least one album tagged as 'punk' by a Jamendo user, sorted by the number of inhabitants of the places they are based near."

```
SELECT DISTINCT ?an ?lat ?long WHERE {
?a a mo:MusicArtist ;
  foaf:based_near ?place ;
  foaf:name ?an ;
  foaf:made ?alb .
?alb tags:taggedWithTag
     <http://dbtune.org/jamendo/tag/jazz> .
?place
  geo:name ?name ;
  geo:population ?population ;
  wgs:lat ?lat ;
  wgs:long ?long } ORDER BY ?population
```

**Example 32.** "Give me the names and the birth dates of guitarists involved in my collection, born in Paris between 1939 and 1945."

```
SELECT ?name ?birth ?description ?person WHERE {
?person
  p:birthPlace <http://dbpedia.org/resource/Paris> ;
  a yago:Guitarist110151760 ;
  p:birth ?birth ;
  foaf:name ?name ;
  rdfs:comment ?description .
  FILTER (LANG(?description) = ‘en’ &&
   ?birth > "1939-01-01"^^xsd:date &&
   ?birth < "1945-01-01"^^xsd:date ) . }
```

**Example 33.** "Give me sets of two different audio files in my collection starting in the same key."

```
SELECT ?audiofile1 ?audiofile2 ?key
WHERE {
?audiofile1 mo:encodes ?signal1 .
?audiofile2 mo:encodes ?signal2 .
?signal1 mo:time [tl:timeline _:tl1] .
?signal2 mo:time [tl:timeline _:tl2] .
?kc1
   a af:KeyChange ;
   af:new_key ?key ;
   event:time [tl:at "PT0S"; tl:timeline _:tl1] .
?kc2
   a af:KeyChange ;
   af:new_key ?key ;
   event:time [tl:at "PT0S"; tl:timeline _:tl2] .
FILTER (?audiofile1 != ?audiofile2)}
```

**Example 34.** "Give me artists in my collection who have been featured more than 10 times in a BBC show."

```
SELECT ?name ?brand ?title ?count
WHERE {
?artist a mo:MusicArtist ;
  foaf:name ?name .
?pc pc:object ?artist ;
  pc:count ?count .
?brand a po:Brand ;
  pc:playcount ?pc ;
  dc:title ?title .
FILTER (?count>10)}
```

Other examples of queries that can be issued to GNARQL are the following ones.

- "Give me artists in my collection ordered by murder rates in their city" (using our Riese dataset of European statistical information [HRH08a, HRH08b])

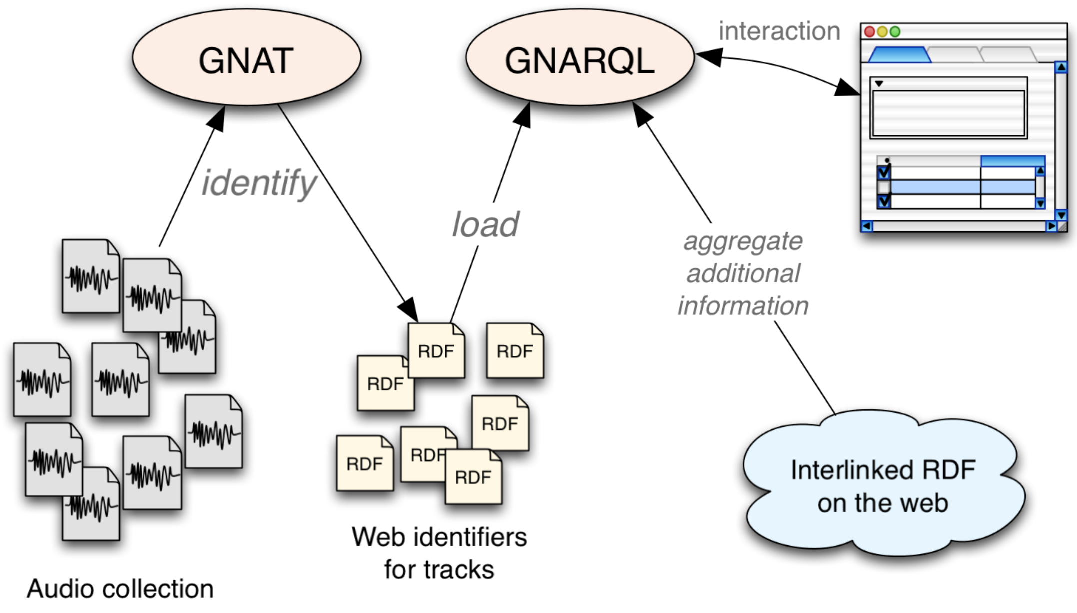

GNAT
GNARQL
interaction
identify
load
aggregate additional information
RDF
Audio collection
Web identifiers for tracks
Interlinked RDF on the web

Figure 8.1: Management of personal music collections using GNAT and GNARQL

- "Give me artists in my collection who were born within 10 miles of my birth place"
- "Give me structural segments of this track (e.g. chorus, verse, etc.)"

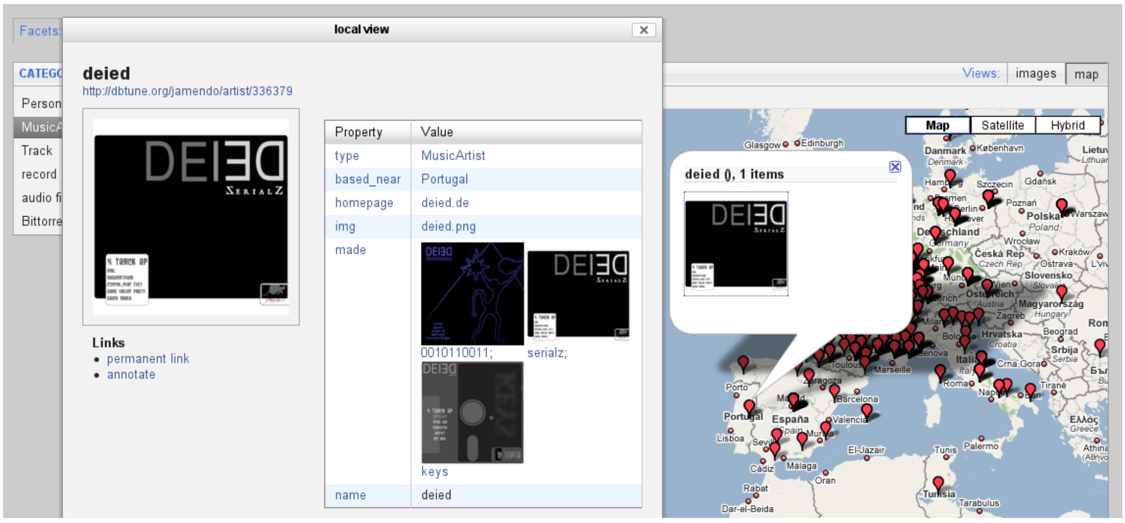
### 8.1.3 User interfaces

User interfaces can then be built on top of the SPARQL end-point GNARQL provides. The overall interactions between GNAT, GNARQL and user interfaces are depicted in Figure 8.1. We integrated the /Facet faceted browsing interface [HvOH06] in GNARQL[3].

Usually, music collection management software only allows the user to browse by album name, track name and artist name. /Facet on top of a GNARQL aggregation provides new ways of interacting with an audio collection. We can plot the artists in a user's collection on a map, as in Figure 8.2. We can then browse a collection by geographical location. We can also plot the birth dates of artists in a user's collection on a timeline, as in Figure 8.3.

Using /Facet on top of a GNARQL aggregation also provides a semantic search capability on top of a user's collection. For example, we can search for resources related to a literal similar to 'punk'. We end up with the results depicted in Figure 8.4. These results correspond to artists, records, tracks, lyrics, performances, with in every case an explanation of how that result relates to 'punk'. For example, a particular record is related to 'punk' because it has a track which 'sounds similar' to another track which has been tagged as punk by a Jamendo user. Clicking on such a record leads us to the page in Figure 8.5. Here we get more information, including cover art, tags, available tracks, artist, Creative Commons licensing information and available download locations. This interface also allows a user to discover more information about his music collection. The user can discover albums he does not yet own made by artists in his collection. He can discover other performances of a particular work and read reviews of these

[3]It remains future work to investigate other user interfaces, such as mSpace [msWRS06].



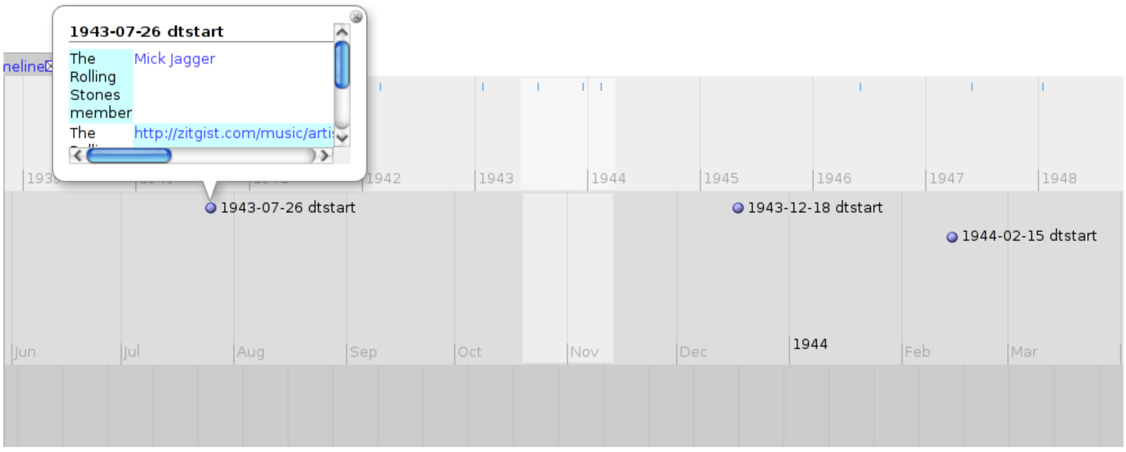
local view

**deied**
http://dbtune.org/jamendo/artist/336379

Links
- permanent link
- annotate

| Property | Value |
|---|---|
| type | MusicArtist |
| based_near | Portugal |
| homepage | deied.de |
| img | deied.png |
| made | 0010110011; serialz; keys |
| name | deied |

Figure 8.2: Plotting the artists in a user's collection on a map. Each pin corresponds to an artist. Here, we clicked on a Portuguese artist to display more information about him, including a depiction.



**1943-07-26 dtstart**

| | |
|---|---|
| The Rolling Stones member | Mick Jagger |
| The | http://zitgist.com/music/arti |

Figure 8.3: Plotting the artists in a user's collection on a timeline using their birth dates. Each dot corresponds to a particular birth.

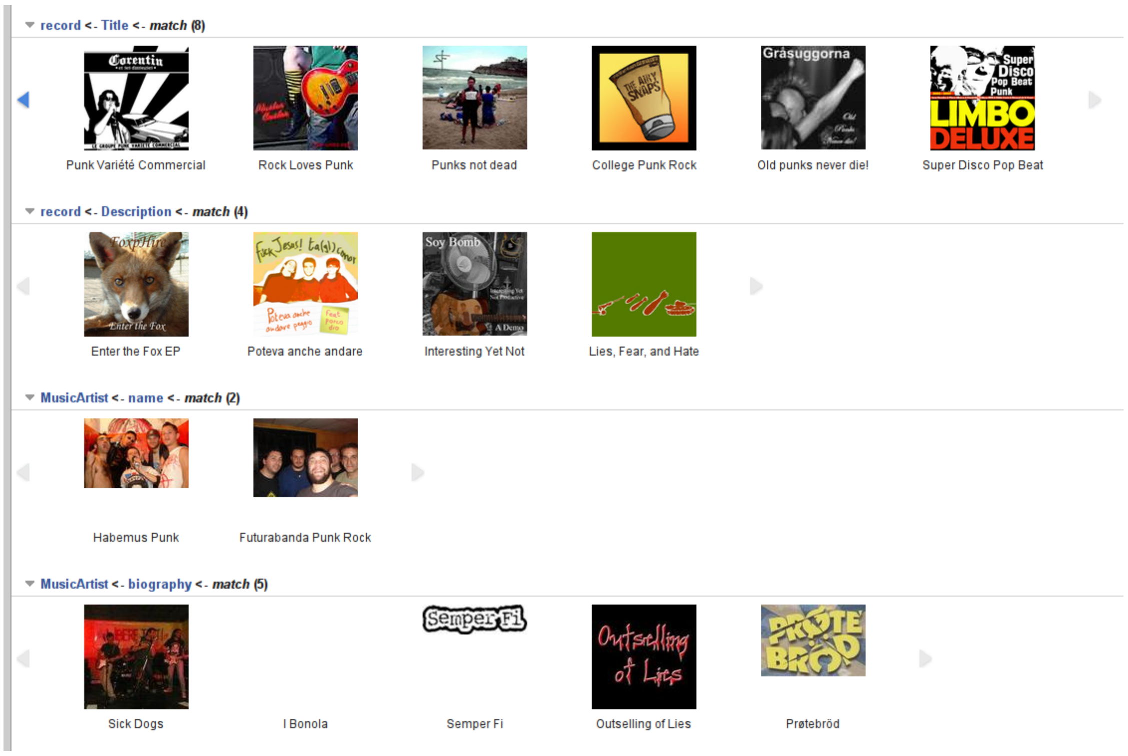


Figure 8.4: Semantic search on top of a user's music collection. First page of results when doing a literal search for 'punk' in a GNARQL aggregation

performances. He can use the tags aggregated from multiple services to browse his collection. He can investigate how artists in his collection relate to his other interests, e.g. politics.

Such an aggregation of information can also be used to display item-specific information, such as a structural segmentation of the audio item, the tonal changes within it, the melody, the different beats. The Sonic Visualiser [CLSB06] provides the ability to visualise such information, as illustrated in Figure 8.6.

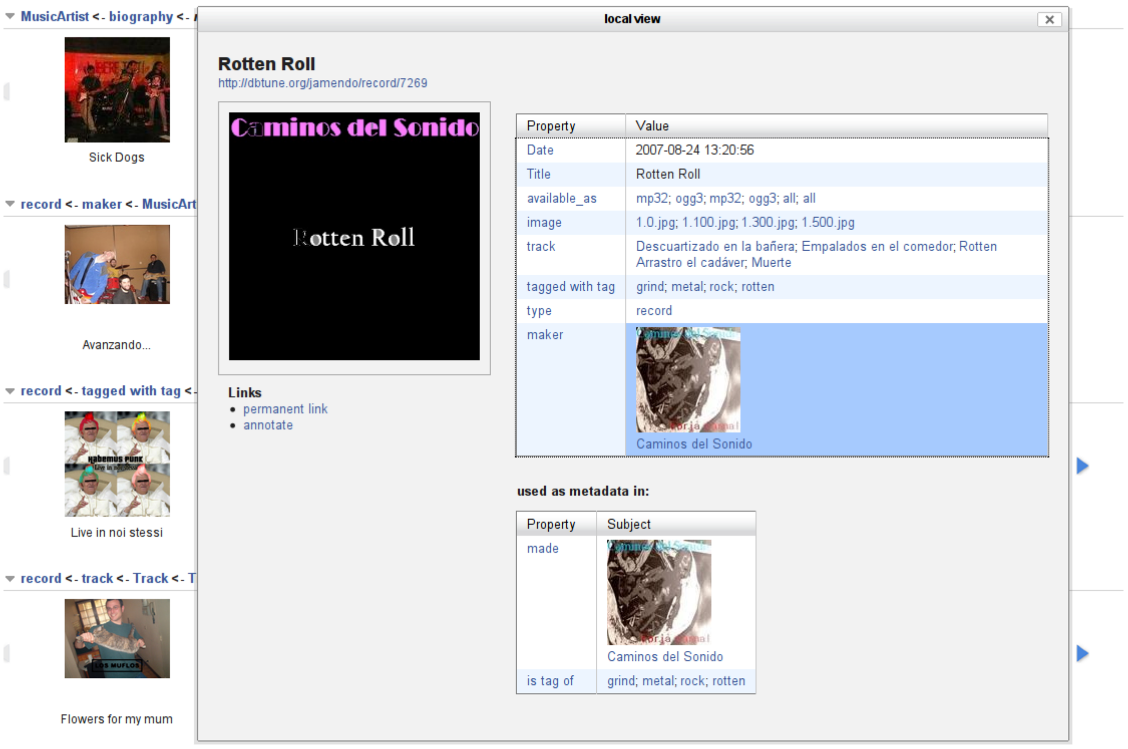
### 8.1.4 Future work on Semantic Web-based music collection management

With only a little further work, it is expected that more useful queries such as "Find gigs by artists similar to my most-played artists which fit within my vacation plan" will be handled also. The range of information being aggregated is not limited to purely musical information — it can also include the user's calendar or vacation plan.

Future work includes the integration of such GNARQL-like aggregators within dedicated music collection management tools, such as Songbird[4]. Another possible extension would be to provide this aggregation as a remote service, whose input is the output of GNAT, and which handles a remote audio collection management interface. Such a set-up would allow us to investigate the interlinks between different personal music collections, to suggest new music based on other similar collections, or to drive a social network.

On-going research on the "semantic desktop" led to the creation of the Nepomuk framework [Con07] within the KDE4 desktop environment. Nepomuk, amongst other things, handles a

[4]http://songbirdnest.com



MusicArtist <- biography <-

Sick Dogs

record <- maker <- MusicArt

Avanzando...

record <- tagged with tag <-

Live in noi stessi

record <- track <- Track <- T

Flowers for my mum

local view

**Rotten Roll**

http://dbtune.org/jamendo/record/7269

Caminos del Sonido

Rotten Roll

Links

- permanent link
- annotate

| Property | Value |
|---|---|
| Date | 2007-08-24 13:20:56 |
| Title | Rotten Roll |
| available_as | mp32; ogg3; mp32; ogg3; all; all |
| image | 1.0.jpg; 1.100.jpg; 1.300.jpg; 1.500.jpg |
| track | Descuartizado en la bañera; Empalados en el comedor; Rotten Arrastro el cadáver; Muerte |
| tagged with tag | grind; metal; rock; rotten |
| type | record |
| maker | Caminos del Sonido |

used as metadata in:

| Property | Subject |
|---|---|
| made | Caminos del Sonido |
| is tag of | grind; metal; rock; rotten |

Figure 8.5: Selecting a particular record in the list of results

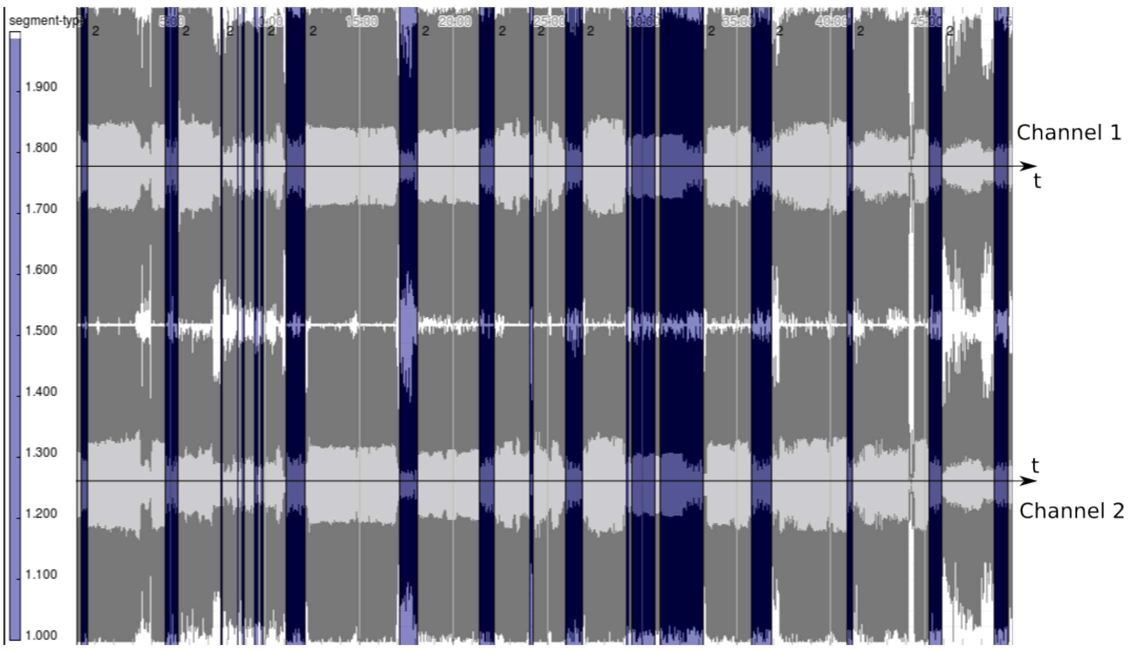


Figure 8.6: Speech/music segmentation of a Tom Robinson BBC 6 Music podcast entry, available at `http://www.bbc.co.uk/radio/podcasts/trintro/`

RDF store shared amongst many different desktop applications. The integration of a GNARQL-like crawler to enrich this RDF store with structured music-related web data would allow audio applications in KDE4 such as Amarok[5] to provide a richer user experience.

A diversity of new use-cases emerges when we also consider keeping track of *production* metadata. Audio creation applications (sequencers, MIDI trackers, audio effects, etc.) could also use this same shared store for production information. When an audio sample is being used, this could be recorded within this store using the corresponding manifestation web identifier, therefore allowing other users to query for tracks that include a sample taken from a particular song. Other users could also query for songs that use a given audio effect, in order for example to listen to a real-world use of them. Also, individual audio takes can be annotated separately, therefore allowing other users to query for information about them and even re-use them if the licensing terms are sufficiently flexible.

It also remains future work to perform a user study similar to Vignoli's study [Vig04] to determine which feature should be predominant in a faceted browsing user interface on top of a GNARQL aggregation.

## 8.2 Enriched podcasts

Podcasts are syndicated multimedia content — content from one web site made available for other sites or applications to re-use. A variety of content is now being "podcasted" to a wide range of devices [Rog05]. Usually, a single entry in an audio podcast consists of a single large audio file. Only a textual description of such entries is made available, usually in HTML. Hence, it is difficult to find a podcast, or to look for parts of a podcast entry: single music tracks, audio commentaries, etc.

Our ZemPod framework [CR08] aims at using our N3-Tr framework and the information sources described in the previous chapters to give more semantics to podcasts, allowing a user to find content within a podcast entry or to use external information for locating particular podcasts.

We first describe common Multimedia web syndication languages in § 8.2.1, and how they can be made available as RDF data in § 8.2.2. We then detail in § 8.2.3 some N3-Tr rules that allows the interpreter described in chapter 7 to automatically enrich Podcasts available in RDF. Finally, we give some examples of how such enriched podcasts can be used in § 8.2.4.

### 8.2.1 Multimedia web syndication

Multimedia web syndication is often performed using XML dialects, such as RSS (Really Simple Syndication: RSS 2.0, Rich Site Summary: RSS 0.91 and RSS 1.0, RDF Site Summary: RSS 1.0), or Atom. An example of a podcast using Media RSS[6] is the following.

```
<rss version="2.0"
  xml:base="http://www.ourmedia.org"
  xmlns:media="http://search.yahoo.com/mrss"
  xmlns:dc="http://purl.org/dc/elements/1.1/">
<channel>
```

[5] http://amarok.kde.org/

[6] mRSS, extending RSS 2.0 with different media types, http://search.yahoo.com/mrss/

```
 <title>Tom Robinson Introducing...</title>
 <link>http://www.bbc.co.uk/6music/shows/tom_robinsons_introducing/</link>
 <description>BBC 6 Music's new album-length podcast</description>
 <language>en</language>
 <item>
  <title>TRIntro: Album Length Podcast 28.10.07</title>
  <link>
 http://downloads.bbc.co.uk/podcasts/6music/trintro/trintro_20071029-0900.mp3
  </link>
  <description>12 free tracks</description>
  <pubDate>Mon, 29 Oct 2007 09:00:00 +0100</pubDate>
  <category>Music</category>
  <dc:creator>Tom Robinson</dc:creator>
  <media:content
   url="http://downloads.bbc.co.uk/podcasts/6music/trintro/
trintro_20071029-0900.mp3"
   fileSize = "47739035" type="audio/mpeg" medium="audio"  duration="2979" />
 </item>
 <item> ... </item>
</channel>
</rss>
```

This piece of mRSS data describes a container for the podcast (the `channel` element). Some data is associated with this podcast: a title, a link to the corresponding web page, a small description and the language used. This podcast holds a number of entries, denoted by the `item` elements, and ordered from the newest to the oldest. The first entry in this podcast has a title, a description, a publication date, a category, a creator, and a link to an audio file. This single audio file holds 12 tracks, all of them with a small introduction made by the presenter. As mentioned in the introduction of this section, it is impossible to access a track within this audio file, or a comment made by the presenter. The user can only search whole podcast entries using the data available in the mRSS feed.

### 8.2.2 Podcasts on the Semantic Web

The previous example can be translated to RDF using the GRDDL specification mentioned in § 2.2.4.4. As a target web ontology for syndication, we consider the use of Atom/OWL [AS06] and our Music Ontology framework described in chapter 3. The example in § 8.2.1 translates to:

```
[] a awol:Feed;
 dc:title "Tom Robinson Introducing...";
 dc:source <http://www.bbc.co.uk/6music/shows/tom_robinsons_introducing/>;
 awol:updated "2007-10-28T09:00:00Z"^^xsd:dateTime;
 awol:author [ a foaf:Person;
    foaf:name "Tom Robinson";
    owl:sameAs _:author1 ;
    ];
 awol:entry [  a awol:Entry;
    awol:author _:author1;
    dc:title "TRIntro: Album Length Podcast 28.10.07";
    dc:description "12 free tracks";
    awol:updated "2007-10-28T09:00:00Z"^^xsd:dateTime;
    awol:content <http://downloads.bbc.co.uk/podcasts/6music/trintro/
trintro_20071029-0900.mp3>;
 ] .

<http://www.ourmedia.org/user/billy2rivers/files/podcast3.mp3> a mo:AudioFile.
```

We describe a resource of type `awol:Feed`, which has attached information (author, title, etc.). This feed links to resources of type `awol:Entry`, which itself links to the audio content.

### 8.2.3 Enriching podcasts with a N3-Tr interpreter

Now, syndicated content can be part of the web of structured data described in the two previous chapters. We consider enriching these podcasts using the N3-Tr interpreter described in chapter 7. We write N3-Tr rules that allow this interpreter to derive and publish more information about these podcasts.

#### 8.2.3.1 Speech/Music segmentation

We first segment audio entries within a podcast into speech parts and music parts. In order to do so, we first need to segment the audio track. We consider identifying a speech/music segmentation process by `sig:speechmusic-segmentation`, associating an audio signal with a list of start and end times of the detected segments in seconds. An example of using this predicate is the following, associating an audio signal with three start and end times.

```
:signal sig:speechmusic-segmentation ((0 35.4) (35.4 223.6) (223.6 260.3)).
```

In the structured representation of `sig:speechmusic-segmentation`, we provide the following N3-Tr rule, detailing how to derive such segmentations by using a particular Vamp plugin [Can] clustering frame-based features [LS08]. We also specify in this N3-Tr rule the specific parameters needed by the plugin to perform an efficient speech/music segmentation (two segment types, at least four seconds each, and using Constant-Q features, as reviewed in § $A.3$).

```
qvp:qm-segmentation a ctr:TabledPredicate.

{?signal sig:speechmusic-segmentation ?segmentation} <=
  {
    (?signal sig:constant-q 2 4) qvp:qm-segmentation ?segmentation
  } .
```

In order to give meaning to these results, we relate them to corresponding Music Ontology terms. This is done through the following N3-Tr rule.

```
{
_:segment
   a af:StructuralSegment;
   event:time [ tl:timeline :tl; tl:start ?start tl:end ?end ]
} <=
 {
   ?signal
     a mo:Signal ;
     mo:time [ tl:timeline :tl] ;
     sig:speechmusic-segmentation ?segmentation .
   (?start ?end) list:in ?segmentation .
 } .
```

The structural segments produced by this N3-Tr rule on a BBC 6 Music podcast are visualised using the Sonic Visualiser as in Figure 8.6. We now want to classify further these structural segments into speech and music. We identify this process by `sig:speechmusic-classify`, linking a particular segment to its speech/music classification. This classification can be derived using the series of tests described in [PT05]. For example, if the maximal mean frequency exceeds 2.4 kHz, the segment is classified as music. For speech signals, the maximal mean frequency is indeed almost always less than 2.4 kHz, while it can be much greater for music segments. We map the results of this speech/music classification to terms within the Audio Features ontology described in § 3.3.4.5.

```
sig:speechmusic-classify a ctr:TabledPredicate .

{ ?segment a af:SpeechSegment } <=
  { ?segment sig:speechmusic-classify sig:speech } .
{ ?segment a af:MusicSegment } <=
  { ?segment sig:speechmusic-classify sig:music } .
```

#### 8.2.3.2 Audio identification

We now consider linking the music parts in this podcast to corresponding web resources in the Musicbrainz dataset described in § 6.3.5. Audio identification (or audio fingerprinting) is a technology for identifying unknown music. An audio fingerprint is a unique and compact digest derived from perceptually relevant aspects of a recording. There are different methods to generate an audio fingerprint. The fingerprint model usually receives a sequence of feature vectors computed on a frame by frame basis, plus some track-level descriptors, such as beats per minute or prominent pitch [BKWW99]. One of the most used methods is Hidden Markov Models, because it can model temporal evolution of audio signals [CBKH05].

We implemented such a fingerprint-based identification within our GNAT software described in § 8.1.1. Indeed, since audio metadata in personal collections is frequently incomplete, inaccurate, or missing entirely, the approach described in § 8.1.1 may not be sufficient. GNAT gives the ability to fall back to a fingerprinting-based approach, using MusicIP's MusicDNS service[7]. Ideally, we could use the fingerprint of an audio file as just another piece of information about the

[7]See `http://www.musicip.com/dns/`

track, incorporating it into the graph mapping approach described in § 6.4.5. However, the MusicDNS service is relatively opaque, and there is no obvious way to uniformly combine fingerprint information and local metadata in a graph mapping approach. A fingerprinting service which exposed the server-side database and the actual process of matching a fingerprint to a database entry could allow for some more sophisticated linkage.

We identify a fingerprint-based lookup by `gnat:fingerprint`, associating an audio signal with a corresponding PUID (MusicDNS's track identifiers). The Musicbrainz dataset also includes PUID information, associated with track resources. Therefore, the following N3-Tr rule provides the linkage from the music segments derived in § 8.2.3.1 to corresponding track resources within the Musicbrainz dataset described in § 6.3.5.

```
gnat:fingerprint a ctr:TabledPredicate .

{ ?signal mo:published_as ?musicbrainztrack } <=
 {
   ?segment
     a af:MusicSegment ;
     event:time ?time .
   ?signal
     mo:time ?time ;
     gnat:fingerprint ?puid .
   ?musicbrainztrack mo:puid ?puid
 } .
```

### 8.2.3.3 Keyword extraction

Finally, we perform keyword extraction on the speech segments derived using the N3-Tr rules in § 8.2.3.1. We use a dictionary of musical terms for this step. We identify this process by `sig:musical-keywords-extraction`, associating an audio signal with the extracted keywords. Such statements can be derived using the CMU Sphinx toolkit[8]. The following N3-Tr rule translates these statements to Music Ontology terms.

```
sig:musical-keywords-extraction a ctr:TabledPredicate .

{ ?segment af:text ?keyword } <=
 {
   ?segment
     a af:SpeechSegment ;
     event:time ?time .
   ?signal
     mo:time ?time ;
     sig:musical-keywords-extraction ?keywords .
   ?keyword list:in ?keywords .
 } .
```

[8]http://cmusphinx.sourceforge.net/html/cmusphinx.php

### 8.2.4 Querying for specific podcasts

We now have N3-Tr rules enriching podcasts available on the Semantic Web. A N3-Tr interpreter can apply them to aggregated podcasts to derive further statements, linking it to the tracks being played in the Musicbrainz dataset and to keywords extracted from its speech parts. The results of such a process can be depicted as in Figure 8.7.

The podcast entry in § 8.2.2 is further described by the following statements, derived by an N3-Tr interpreter.

```
<http://downloads.bbc.co.uk/podcasts/6music/trintro/trintro_20071029-0900.mp3>
   a mo:AudioFile ;
   mo:encodes :signal .
:signal mo:time [tl:duration "PT2979S"; tl:timeline :tl] .
:segment1
   a af:MusicSegment ;
   event:time [tl:start "PT0S"; tl:end "PT13.7S"; tl:timeline :tl] .
:segment2
   a af:SpeechSegment ;
   af:text "podcast","bbc","album-length","introducing","artist","music" ;
   event:time [tl:start "PT13.7S"; tl:end "PT35.75S"; tl:timeline :tl] .
:segment3
   a af:MusicSegment ;
   event:time [owl:sameAs :time3; tl:start "PT35.75S"; tl:end "PT4M44S";
     tl:timeline :tl] .
:signal3
   a mo:Signal ;
   mo:published_as <http://dbtune.org/musicbrainz/resource/track/
f5e93176-0829-4823-a5be-3333e226824b> ;
   mo:time :time3 .
:segment4
   a af:SpeechSegment ;
   af:text "love","dance-floor","arctic monkeys","rock","girl","fratellis" ;
   event:time [tl:start "PT4M44S"; tl:end "PT5M21S"; tl:timeline :tl] .
:segment5
   a af:MusicSegment ;
   event:time [owl:sameAs :time5; tl:start "PT5M21S"; tl:end "PT7M39S";
     tl:timeline :tl] .
:signal5
   a mo:Signal;
   mo:published_as <http://dbtune.org/musicbrainz/resource/track/
0d069749-6d54-438f-b1dd-12c9be97416f> ;
   mo:time :time5 .
:segment6
   a af:SpeechSegment ;
   af:text "go team","dance","sextet" ;
```

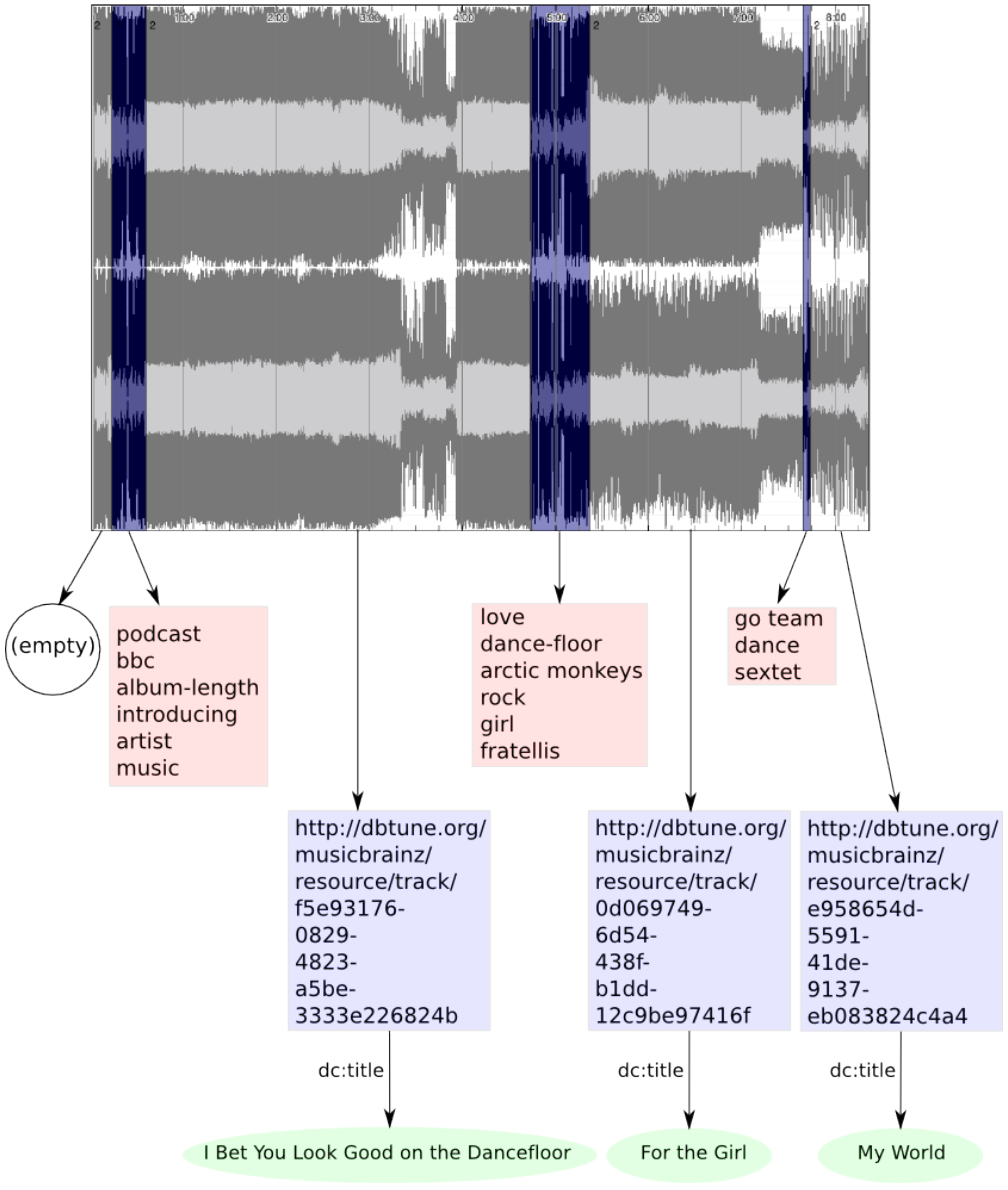

2
1:00
2:00
3:00
4:00
5:00
6:00
7:00
8:00
(empty)
podcast
bbc
album-length
introducing
artist
music
love
dance-floor
arctic monkeys
rock
girl
fratellis
go team
dance
sextet
http://dbtune.org/musicbrainz/resource/track/f5e93176-0829-4823-a5be-3333e226824b
http://dbtune.org/musicbrainz/resource/track/0d069749-6d54-438f-b1dd-12c9be97416f
http://dbtune.org/musicbrainz/resource/track/e958654d-5591-41de-9137-eb083824c4a4
dc:title
dc:title
dc:title
I Bet You Look Good on the Dancefloor
For the Girl
My World

Figure 8.7: Depiction of the statements derived by a N3-Tr interpreter using the rules defined in § 8.2.3 and a particular Podcast entry. The blue segments correspond to detected speech segments. The grey segments correspond to detected music segments. The red boxes correspond to the results of the speech recognition process, the blue boxes correspond to the results of the fingerprinting process.

```
   event:time [tl:start "PT7M39S"; tl:end "PT7M43S"; tl:timeline :tl] .
:segment7
   a af:MusicSegment ;
   event:time [owl:sameAs :time7; tl:start "PT7M43S"; tl:end "PT10M22S";
     tl:timeline :tl] .
:signal7
   a mo:Signal;
   mo:published_as <http://dbtune.org/musicbrainz/resource/track/
e958654d-5591-41de-9137-eb083824c4a4> ;
   mo:time :time7 .
```

The podcast entry is associated to a timeline `:tl`, which is decomposed in music and speech segments. The music segments are linked to the corresponding Musicbrainz track identifiers, which lead to further details about the tracks played (artists involved in the track, releases on which it appears, etc.), but also provide links to other datasets. The breadth of information available to search for a particular podcast entry is dramatically increased. As we have access to the segments withing a podcast entry, we can also directly access specific parts of it. To illustrate that, we consider the following queries.

- "Give me all the podcasts that featured an Arctic Monkeys' song";
- "Give me podcast entries associated to the words 'punk' and 'rock' ";
- "At what time does that song start, in this podcast entry?";
- "Give me podcasts that only feature tracks performed by English artists";
- "Give me podcast entries that feature an artist involved in that political movement";
- "Give me two podcast entries that feature artists that are married to each others".

## 8.3 Data-rich music recommendation

The interlinked structured data described in chapters 6 and 7 can also be used for automating the process of recommending musical resources (artists, tracks, works, performances, etc.) to users.

We first overview the two main paradigms in music recommendation in § 8.3.1. We then explain how available structured web data can be used for track recommendations in § 8.3.2, for event recommendations in § 8.3.3 and for artist recommendations in § 8.3.4.

### 8.3.1 Current music recommendations systems

Current music recommendation systems can be clustered in two main categories: those using collaborative filtering and those using content-based similarities.

Collaborative filtering systems recommend items to a user based on the stated tastes of other users. For example, a user $u_2$ might be recommended a song $s_2$ if he likes a song $s_1$, and that a user $u_1$ likes both $s_1$ and $s_2$. Usually, music recommendations service based on such a methodology use a closed set of information, gathered through a particular service, e.g. listening habits for last.fm or consumer behaviours for Amazon. Some systems adapt similar ideas in open information environments, such as in [CF00], where structured taste information is extracted from web documents. Collaborative filtering can also be enhanced by integrating contextual data, reflecting the user's interests at a particular moment [Hay03]. However, collaborative filtering does not tackle the problem of finding items within the long tail of music production [CC08] — the items for which the amount of taste data is limited. Collaborative filtering is only suitable when a large number of users have heard the music and expressed their interest in it.

A recent trend for music recommendation is to use automated content analysis to drive recommendations, by modelling musical audio similarity [Pam06]. For example, a particular song might be recommended to a user because its rhythm is similar to a song mentioned in his preferences. A system based on such a methodology will therefore be able to recommend items within the long tail [CC08], as the system has no notion of 'popularity' — it will be able to recommend artists for which the amount of taste data is limited. Similar results are obtained by using an expert-based approach, where qualified users relate songs by musical criteria, such as Pandora and the Music Genome Project[9].

Several works combine these two methodologies (collaborative filtering and content-based) to achieve better recommendations [BS97, YGK$^+$06]. Content-based recommendation can indeed help with the cold start problem of collaborative filtering, where we wish to recommend items that no users has yet rated.

We believe these two approaches (collaborative filtering and content-based similarities) can be improved by using interlinked web data in multiple domains. Starting from a user's profile available as structured web data, as in the FOAFing-the-music project [CRH05], we gather a set of web identifiers, corresponding to explicit tastes (e.g. 'I like that band', 'I am involved in that movement') or implicit tastes (e.g. 'I have been listening to that band a lot, recently', 'I went to that conference, which primary topic was that movement'). Starting from these web identifiers, we create a path in structured web data leading towards the recommended resource. Such a path can be used to *explain* the recommendations to the user (e.g. 'We recommended you this artist's event because he used to play saxophone with that other artist you like'). We now study three ways of deriving such recommendation paths.

### 8.3.2 Track recommendations

As described in § 7.3.2, our SBSimilarity Semantic Web service exposes content-based similarity statements. The SBSimilarity service also exposes RDF link to resources within the Musicbrainz dataset described in § 6.3.5, Amazon purchase pages and audio previews.

We designed a simple user interface for this service which ties into iTunes, as illustrated in Figure 8.8. This user interface demonstrates how a user's audio collection can easily provide entry points for exploring the growing body of data on the Semantic Web. It also shows how computation can be hidden behind Semantic Web resources to provide user agents with a uniform data access mechanism for both pre-existing information and information that is computed on-demand. This information may be either human-authored (coming from an editorial database such as Musicbrainz, for example) or automatically extracted from multimedia content (as in our similarity example).

In this application, the track recommendations start from items in a personal audio collection. Then, the user specifies himself the path he wants in order to discover new tracks, through the

[9]`http://www.pandora.com/`

Tabulator interface [BLCC+06]. He can either use solely content-based similarity statements derived by the SBSimilarity service or use Musicbrainz linked data (e.g. relationships between artists).

### 8.3.3 Event recommender

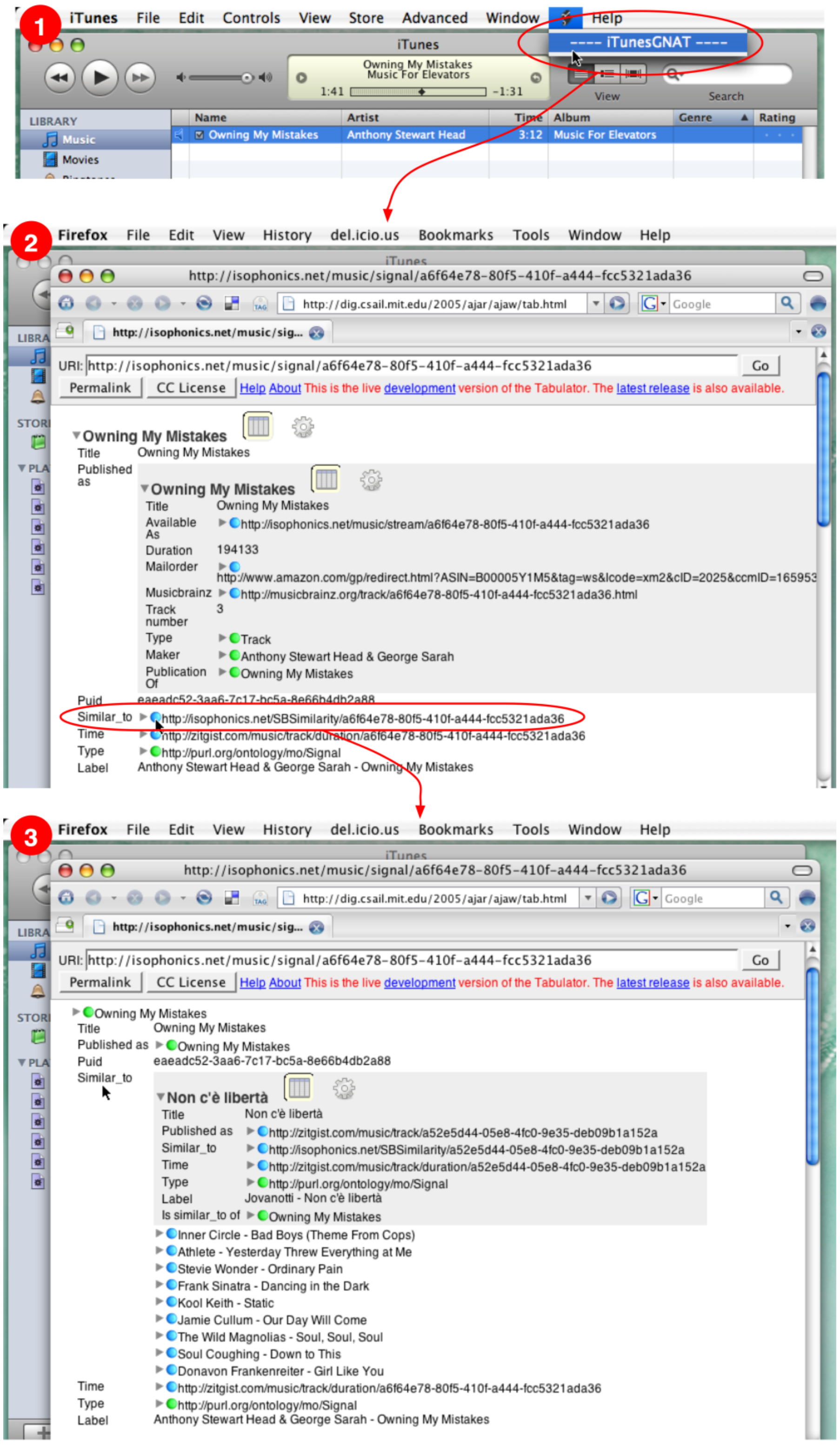
As in the Foafing-the-music project [CRH05], we can use structured FOAF user profiles [BM07] made available on the Web to drive recommendations. A FOAF profile can include social networking information (using the `foaf:knows` property and the corresponding sub-properties), interests (using the `foaf:topic_interest` property), online accounts on different web sites, different identities on different services, activities on these services and geo-location data.

```
<http://moustaki.org/foaf.rdf#moustaki>
   a foaf:Person ;
   foaf:name "Yves Raimond" ;
   foaf:nick "moustaki" ;
   foaf:openid <http://moustaki.myopenid.com> ;
   foaf:weblog <http://blog.dbtune.org/> ;
   foaf:based_near [ a geo:Point; wgs:lat "51.522894"; wgs:long "-0.040834" ] ;
   foaf:knows <http://danbri.org/foaf.rdf#danbri> ;
   foaf:knows <http://chrissutton.org/me> ;
   foaf:knows <http://beckr.org#chris> ;
   owl:sameAs <http://dbtune.org/last-fm/moustaki> ;
   owl:sameAs <http://dblp.l3s.de/d2r/resource/authors/Yves_Raimond> ;
   foaf:topic_interest <http://dbpedia.org/resource/Semantic_Web> ;
   foaf:topic_interest <http://dbpedia.org/resource/Punk_Rock> ;
   foaf:topic_interest <http://dbpedia.org/resource/Johann_Sebastian_Bach> .
```

This FOAF profile links a web identifier representing a person to an equivalent web identifier on the AudioScrobbler service described in § 6.3.4, giving access to the listening habits of this person (which tracks have been listened to and when). We can use such a profile as a basis for recommendations.

The AudioScrobbler RDF service described in § 6.3.4 includes recommended events. For example, it publishes the following data for the above profile.

```
<http://dbtune.org/last-fm/moustaki>
  lfm:recommendation [
    a mo:Performance;
    dc:title "Rebellion London on 13 Dec 2008";
    event:place [
      dc:title "The Forum, West Molesey, KT81NS, UK";
      wgs:long "-0.364315";
      wgs:lat "51.402759" ];
    event:time [a tl:Interval; tl:start "2008-12-13"];
    foaf:homepage <http://www.last.fm/event/696151>;
  ] .
```


1
iTunes File Edit Controls View Store Advanced Window Help
---- iTunesGNAT ----
iTunes
Owning My Mistakes
Music For Elevators
1:41
-1:31
View
Search
LIBRARY
Music
Movies
2
Firefox File Edit View History del.icio.us Bookmarks Tools Window Help
http://dig.csail.mit.edu/2005/ajar/ajaw/tab.html
Google
http://isophonics.net/music/sig...
3

| Name | Artist | Time | Album | Genre | Rating |
|---|---|---|---|---|---|
| Owning My Mistakes | Anthony Stewart Head | 3:12 | Music For Elevators | | |

http://isophonics.net/music/signal/a6f64e78-80f5-410f-a444-fcc5321ada36

URI: http://isophonics.net/music/signal/a6f64e78-80f5-410f-a444-fcc5321ada36 Go

Permalink CC License Help About This is the live development version of the Tabulator. The latest release is also available.

Owning My Mistakes
Title Owning My Mistakes
Published as
  Owning My Mistakes
  Title Owning My Mistakes
  Available As http://isophonics.net/music/stream/a6f64e78-80f5-410f-a444-fcc5321ada36
  Duration 194133
  Mailorder http://www.amazon.com/gp/redirect.html?ASIN=B00005Y1M5&tag=ws&lcode=xm2&cID=2025&ccmID=165953
  Musicbrainz http://musicbrainz.org/track/a6f64e78-80f5-410f-a444-fcc5321ada36.html
  Track number 3
  Type Track
  Maker Anthony Stewart Head & George Sarah
  Publication Of Owning My Mistakes
Puid eaeadc52-3aa6-7c17-bc5a-8e66b4db2a88
Similar_to http://isophonics.net/SBSimilarity/a6f64e78-80f5-410f-a444-fcc5321ada36
Time http://zitgist.com/music/track/duration/a6f64e78-80f5-410f-a444-fcc5321ada36
Type http://purl.org/ontology/mo/Signal
Label Anthony Stewart Head & George Sarah - Owning My Mistakes

http://isophonics.net/music/signal/a6f64e78-80f5-410f-a444-fcc5321ada36

URI: http://isophonics.net/music/signal/a6f64e78-80f5-410f-a444-fcc5321ada36 Go

Permalink CC License Help About This is the live development version of the Tabulator. The latest release is also available.

Owning My Mistakes
Title Owning My Mistakes
Published as Owning My Mistakes
Puid eaeadc52-3aa6-7c17-bc5a-8e66b4db2a88
Similar_to
  Non c'è libertà
  Title Non c'è libertà
  Published as http://zitgist.com/music/track/a52e5d44-05e8-4fc0-9e35-deb09b1a152a
  Similar_to http://isophonics.net/SBSimilarity/a52e5d44-05e8-4fc0-9e35-deb09b1a152a
  Time http://zitgist.com/music/track/duration/a52e5d44-05e8-4fc0-9e35-deb09b1a152a
  Type http://purl.org/ontology/mo/Signal
  Label Jovanotti - Non c'è libertà
  Is similar_to of Owning My Mistakes
  Inner Circle - Bad Boys (Theme From Cops)
  Athlete - Yesterday Threw Everything at Me
  Stevie Wonder - Ordinary Pain
  Frank Sinatra - Dancing in the Dark
  Kool Keith - Static
  Jamie Cullum - Our Day Will Come
  The Wild Magnolias - Soul, Soul, Soul
  Soul Coughing - Down to This
  Donavon Frankenreiter - Girl Like You
Time http://zitgist.com/music/track/duration/a6f64e78-80f5-410f-a444-fcc5321ada36
Type http://purl.org/ontology/mo/Signal
Label Anthony Stewart Head & George Sarah - Owning My Mistakes

Figure 8.8: A modified version of the GNAT tool described in § 8.1.1 was wrapped as a plugin for Apple's iTunes media player. A user may click a menu item (1) to identify the currently-playing track, and display available information in the Tabulator [BLCC+06] (2). This information includes the advertisement resource for similar tracks, which, when accessed, will trigger the calculation and retrieve the actual similarity statements (3). The user can then browse the Semantic Web to find more information about the tracks, preview their audio, and perhaps choose to buy them from Amazon.

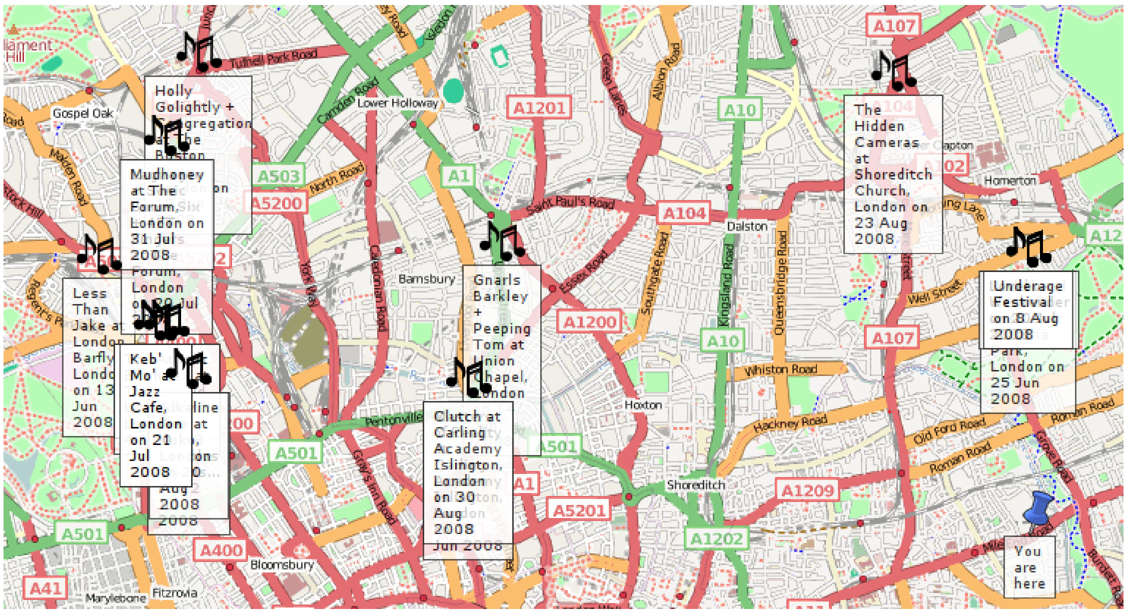


Figure 8.9: Displaying nearby recommended musical events based on a user profile using DBpedia Mobile and our recommended events RDF service. The blue pin corresponds to the location of the mobile device. The 'musical notes' pins correspond to recommended events. Clicking on one of them allows the user to access more information about the event and to book tickets.

Now, the user's FOAF profile and the AudioScrobbler RDF service can be used within DBpedia Mobile [BB08] in order to display recommended events near the geo-location of the user, alongside other recommended sights or items. This geo-location can be made available in his FOAF profile (as above), or can be captured by his mobile device. In Figure 8.9, we used a mobile device with a geo-location support. We went to the DBpedia mobile web site and entered our web identifier to display nearby recommended event in a time frame of three months.

### 8.3.4 Artist recommender

Using FOAF user profiles such as the one in § 8.3.3, we can also exploit the distributed social network it defines to recommend artists to a particular person. A simple collaborative filtering approach is performed by the following SPARQL query.

```
SELECT ?artist
WHERE {
  <$uri> foaf:knows [ foaf:topic_interest ?artist ] .
  ?artist a mo:MusicArtist .
}
```

where `$uri` is the user's web identifier. An artist is recommended if the user has someone in his social network who is interested in this artist. We can also use the following SPARQL query to derive recommendations using both collaborative filtering and content-based similarity statements derived by the SBSimilarity service described in § 7.3.2.

```
SELECT ?artist2
WHERE {
  <$uri> foaf:knows [ foaf:topic_interest ?artist ] .
```

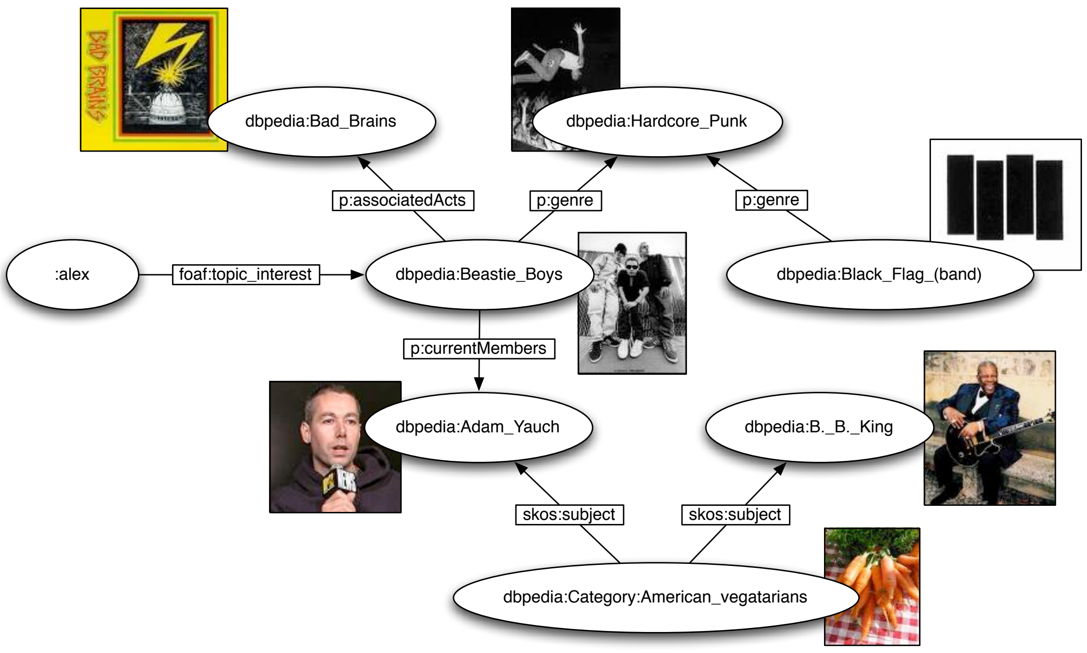


Figure 8.10: Relationships between various artists

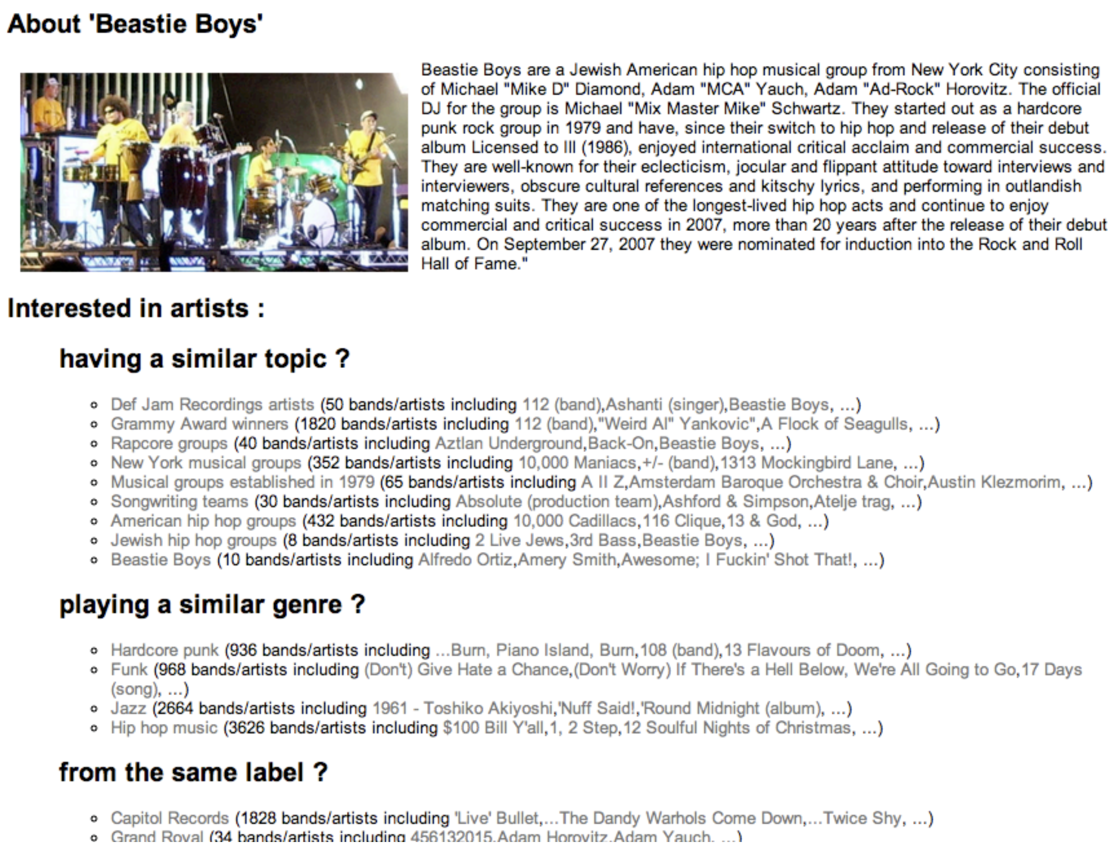
```
  ?artist a mo:MusicArtist; foaf:made ?track1.
  ?track1 mo:similar_to ?track2.
  ?artist2 foaf:made ?track2.
}
```

The user has someone in his social network who is interested in an artist that made a song texturally similar to a song made by the recommended artist. Of course, we can go a lot further using the interlinked datasets mentioned in chapters 6 and 7, by following different sorts of paths from a user to an artist. Moreover, these paths can be used to give an explanation of the recommendation. For example, an artist may be recommended because he played in the same venue as an artist mentioned in the user's profile. Some examples of such paths are depicted in Figure 8.10.

For such complex paths, the challenge is to find relevant paths between the user and the recommended artist. A first step is to display all possible paths, and let the user choose amongst them, as in § 8.3.2. We designed such an artist recommender application [PR08], as depicted in Figure 8.11. In a personal collection management setting, such paths can be explored using /Facet and GNARQL, as described in § 8.1.3. A previous study on user interaction with personal music collections [Vig04] suggests that such a feature (artist recommendations within a personal collection management tool) is liked by the users.

Interesting future work would be to derive automatically such paths, from other explicit (stated tastes) or implicit (listening habits, personal music collection) interests of the user. For example, if the user is interested in politics, he might be interested in a recommendation based on the involvement of an artist in a political movement. If the user is interested in graphic arts, he might be interested in a recommendation based on similar cover arts.

Moreover, each recommendation is associated to a web path, leading towards the recommended resource. As mentioned earlier, such a path can be used to explain the recommendations to the



**About 'Beastie Boys'**

Beastie Boys are a Jewish American hip hop musical group from New York City consisting of Michael "Mike D" Diamond, Adam "MCA" Yauch, Adam "Ad-Rock" Horovitz. The official DJ for the group is Michael "Mix Master Mike" Schwartz. They started out as a hardcore punk rock group in 1979 and have, since their switch to hip hop and release of their debut album Licensed to Ill (1986), enjoyed international critical acclaim and commercial success. They are well-known for their eclecticism, jocular and flippant attitude toward interviews and interviewers, obscure cultural references and kitschy lyrics, and performing in outlandish matching suits. They are one of the longest-lived hip hop acts and continue to enjoy commercial and critical success in 2007, more than 20 years after the release of their debut album. On September 27, 2007 they were nominated for induction into the Rock and Roll Hall of Fame."

**Interested in artists :**

**having a similar topic ?**

- Def Jam Recordings artists (50 bands/artists including 112 (band),Ashanti (singer),Beastie Boys, ...)
- Grammy Award winners (1820 bands/artists including 112 (band),"Weird Al" Yankovic",A Flock of Seagulls, ...)
- Rapcore groups (40 bands/artists including Aztlan Underground,Back-On,Beastie Boys, ...)
- New York musical groups (352 bands/artists including 10,000 Maniacs,+/- (band),1313 Mockingbird Lane, ...)
- Musical groups established in 1979 (65 bands/artists including A II Z,Amsterdam Baroque Orchestra & Choir,Austin Klezmorim, ...)
- Songwriting teams (30 bands/artists including Absolute (production team),Ashford & Simpson,Atelje trag, ...)
- American hip hop groups (432 bands/artists including 10,000 Cadillacs,116 Clique,13 & God, ...)
- Jewish hip hop groups (8 bands/artists including 2 Live Jews,3rd Bass,Beastie Boys, ...)
- Beastie Boys (10 bands/artists including Alfredo Ortiz,Amery Smith,Awesome; I Fuckin' Shot That!, ...)

**playing a similar genre ?**

- Hardcore punk (936 bands/artists including ...Burn, Piano Island, Burn,108 (band),13 Flavours of Doom, ...)
- Funk (968 bands/artists including (Don't) Give Hate a Chance,(Don't Worry) If There's a Hell Below, We're All Going to Go,17 Days (song), ...)
- Jazz (2664 bands/artists including 1961 - Toshiko Akiyoshi,'Nuff Said!,'Round Midnight (album), ...)
- Hip hop music (3626 bands/artists including $100 Bill Y'all,1, 2 Step,12 Soulful Nights of Christmas, ...)

**from the same label ?**

- Capitol Records (1828 bands/artists including 'Live' Bullet,...The Dandy Warhols Come Down,...Twice Shy, ...)
- Grand Royal (34 bands/artists including 456132015,Adam Horovitz,Adam Yauch, ...)

Figure 8.11: Artist recommendation using arbitrary paths in structured web data

user. Although future work needs to evaluate that, we believe such explanations, instead of the usual 'if you like *a* you may like *b*', may improve the user's trust in a recommendation.

## 8.4 Summary

In this section, we described three different applications of the interlinked structured web data described in chapters 6 and 7. We described a personal music collection management tool, aggregating structured data and providing new ways to interact with a collection. We described a framework for enriching syndicated audio content, in order to allow users to find specific parts of such content or to find content according to external criteria. Finally, we describe our work on providing recommendations of different types of resources (tracks, events and artists), using structured web data as a basis.

All these applications show the richness of the applications that can be built on top of the distributed music-related information environment we developed in this thesis. This information, human-authored or automatically generated, can be used to drive collection management applications, visualisation applications, enhanced music access applications, or recommendation applications. Moreover, this information environment has also successfully been applied in other contexts, such as in the EASAIER digital music library system [SRB+07].

# Part IV

# Conclusion and future work

In this thesis we have addressed the distribution of music-related information. We have presented a web-based information environment gathering a wide range of music-related information from multiple sources, including music analysis tools and large online databases. Information available within this environment can be used by automated agents, processing it to derive new information, or performing specific tasks on behalf of a human user, for example to help organising a personal music collection or investigating a particular musical item.

We now summarise the main outcomes of the research within this thesis, and give our perspectives for future work.

## Conclusion

In order to provide anchor points for music information on the Web, we defined an ontology of the music domain in chapter 3. Our ontology tackles editorial information, complex music production workflows, and a mechanism for temporal annotations. Such temporal annotations can be used for a wide range of purpose, including automated annotations by music analysis tools. Our ontology can be used alongside other web ontologies, to deal with e.g. relationships between persons, instrument taxonomies or geographic information. Our ontology does not enforce any particular domain boundaries, and can be used in many different contexts, as illustrated in chapter 6. We compared our ontology with related web ontologies, and showed that ours is the most expressive.

We designed in chapter 4 a methodology for validating music ontologies with regards to real-world music-related information needs. This methodology evaluates how well a system backed by our ontology would answer a set of user queries. We evaluated the performances of our ontology according to this methodology, for two distinct datasets of user needs. We concluded that our ontology covers more than 70% of casual user needs, and more than 78% of music library user needs.

This ontology framework allows us to integrate a wide range of music-related information on the Web. This information can be interpreted by automated agents for different purposes. One of these purposes is to derive new information from existing information. We specified in chapter 5 a knowledge representation framework, N3-Tr, enabling this use-case. This representation framework allows automated agents to derive new music-related information by combining information issued from multiple sources, including music analysis tools. This derived information is published on the Web, allowing other agents to process it. This means that further information can be built on top of published workflows and datasets, and end-user tools can be directly built on top of this information environment.

In order to show that such web-based technologies provide a good basis for distributing music-related information, we applied them to create a large and dynamic web of musical information.

In chapter 6, we used our ontology of the music domain to integrate a vast array of music information sources, which led us to a distributed database holding more than 15 billion interlinked music-related facts. To ease the task of interlinking different music-related information sources, we designed an efficient algorithm, browsing web datasets to automatically draw links between resources that identify the same object.

We described our N3-Tr agent in chapter 7, aggregating such web information, musical content and N3-Tr workflows to provide an on-demand access to automatically derivable information. This derived information can be used by further N3-Tr agents, or by other applications for different purposes.

In chapter 8, we described several use-cases of this distributed and dynamic information environment. The first use-case consists of aggregating web information describing a user's collection to provide innovative ways of interacting with the collection, e.g. plotting the collection on a map, on a timeline, discovering relationships between artists, or recommending titles the user does not yet own. The second use-case provides an enhanced access to syndicated audio content. We designed a set of N3-Tr formulæ, segmenting syndicated audio content in speech and music parts, extracting keywords from the speech parts and linking the music parts to the corresponding identifiers within the Musicbrainz dataset described in § 6.3.5. Such an enrichment of the syndicated audio content enables new ways of accessing it. For example, a user can query for all segments corresponding to a track being played, for all entries associated to a set of keywords, or for all the entries that involve an artist based in a particular country. Finally, our third use-case describes the first steps towards systems that use such a wide range of interlinked data to derive artists, events or tracks recommendations.

These use-cases show that the web-based approach taken in this thesis allows us to be far more flexible about what information is published and how it is reused than in traditional music information management systems. As more information is added to the Web, the possibilities for combination and reuse of information should allow for some exciting and unforeseen uses to emerge.

## Future work

Throughout the thesis, we highlighted a number of possible future directions. We now summarise the most important ones.

**Ontology evaluation** The evaluation methodology designed and applied in chapter 4 must be performed at the conceptual layer in order to give relevant results. We only used a fully automated evaluation at the lexical layer to provide some insights for comparison with related representation frameworks. The different evaluations we performed at the conceptual layer all include a manual annotation step. It remains future work to develop a fully automated evaluation method for music ontologies.

Moreover, it would be interesting to gather a large number of real-world user needs from a set of music libraries and to perform our evaluation on this dataset. It would also be interesting to perform our evaluation on other datasets, e.g. queries drawn by professional musicians or queries drawn by sound engineers.

**Extending the ontology** Our evaluation led us to identify some aspects of music-related information that our ontology framework does not capture. It remains future work to extend our ontology to tackle these different aspects, including partial characterisation of lyrics, emotions and descriptions of related media.

**Uncertainty handling** In particular, handling uncertainty within our ontology leads to deeper research problems. How can we tackle uncertainty in queries issued to an aggregation of structured web data? A similar issue is the one we highlighted in § 5.2.6. How can we interpret and propagate the confidence attached to a set of statements?

**Random predicates** Another related issue is to consider random predicates within the N3-Tr determinism categories defined in § 5.2.3.2. For example, if a predicate `:p` involves a random process, for a given set of inputs it can give a potentially infinite number of outputs, determined by a probability distribution `:p(outputs|inputs)`. Typically, Monte Carlo-based algorithms would be modelled as such random predicates. For now, the interpreter described in chapter 7 handles such predicates in a simplistic way. The interpreter has two modes, `old` and `new`. In the `new` mode, the interpreter derives new results every time a random predicate is evaluated. In the `old` mode, the interpreter retrieves previous evaluations. Future work includes the integration of random predicates within our N3-Tr logic.

**Integrating Description Logics and N3-Tr** Another important area of future work, highlighted in § 5.4.4, is to fully integrate Description Logics and N3-Tr. Description Logics Programs [GHVD03] provide the first steps to achieve such an integration. An ontology can be translated to DLP axioms, which correspond to non-updating N3-Tr rules. For example, we have N3-Tr rules corresponding to the ontology axioms specified in chapter 3. These N3-Tr rules can be used alongside other N3-Tr workflows by the N3-Tr agent described in chapter 7.

**Semantic Web services** Our N3-Tr knowledge representation framework and our publishing framework are not specific to music-related workflows. As mentioned in § 5.4.2, N3-Tr could be used in a Semantic Web services context by including a new item in our transition oracle corresponding to the deletion of an RDF triple. Applying N3-Tr in a Semantic Web services context would require a shift from a service-oriented view to an information-oriented view. What is required by the user is the information delivered by the service, not the service itself. This approach is promising, as SPARQL can then be used to access dynamically generated information.

**Visualisation tools** Apart from the GNARQL tool presented in § 8.1, we are lacking in applications able to provide visualisations of music-related information available on the Web. Future work includes the creation of information visualisers, allowing us to map particular concepts to a relevant visualisation. The Sonic Visualiser [CLSB06] already allows us to visualise timeline information available on the Web, such as structural segmentations or key changes.

Moreover, a graphical environment for the creation, edition and publication of N3-Tr rules is also needed. Recent research [OGA$^+$06, MPT07] on the design of graphical applications for editing workflows would provide a good basis for such an environment.

**Semantic Desktop** An integration of our work within a "Semantic Desktop" environment such as Nepomuk [Con07] is also an interesting direction for future work. Nepomuk provides an RDF store, with which multiple desktop applications can interact (e.g. music player, file manager, audio editor). By using a similar approach as in our GNAT and GNARQL tools, we could aggregate information about the different audio items available on the user's desktop. All the user's applications would then be able to use this information, from personal collection management tools to audio production applications. The integration of aggregated music-related information and other desktop information, such as the user's calendar or vacation plan, would enable queries such as "Find gigs by artists similar to my most-played artists which fit within my vacation plan".

**Producing information in creative music production** Our ontology can be used to capture a number of events pertaining to music production. Capturing and publishing this information during the production process would lead to interesting use-cases. For example, when recording a particular signal, we can keep track of the position of the microphone, the type of microphone being used, the audio effects used, etc. Then, finding an audio signal in the resulting pool of audio signals can be made easier. Moreover, this production information can be shared across different studio applications. Fazekas et al. [FRS08] are starting to explore a software architecture enabling the capture of rich ontology-based information during the production process.

**Data-rich recommendation** An interesting direction for future work is also to improve the recommender systems described in § 8.3. In particular, automatically deriving recommendation paths in structured web data is a promising area of work. We described in this thesis a large web of interlinked music-related data, and there is a large amount of paths leading from a user profile (including e.g. a favourite artist or other interests) to potentially recommendable artists. We need to take into account as much information as possible to derive relevant recommendations. For example, if the user is interested in politics, he might be interested in a recommendation based on the membership of artists in political movements. User studies would be needed, to assess which paths in structured web data are going to lead to relevant recommendations.

# Bibliography

[AB06] H. Alani and C. Brewster. Metrics for ranking ontologies. In *Proceedings of the 4th Int. Workshop on Evaluation of Ontologies for the Web*, 2006.

[ABL$^+$07] S. Auer, C. Bizer, J. Lehmann, G. Kobilarov, R. Cyganiak, and Z. Ives. DBpedia: A nucleus for a web of open data. In *Proceedings of the International Semantic Web Conference*, Busan, Korea, November 11-15 2007.

[AF94] J. F. Allen and G. Fergusson. Actions and events in interval temporal logic. Technical Report TR521, University of Rochester, Computer Science Department, 1994. Available at `http://citeseer.ist.psu.edu/allen94actions.html`. Last accessed February 2008.

[All83] James F. Allen. Maintaining knowledge about temporal intervals. *Artificial Intelligence and Language Processing*, 26:832–843, 1983.

[Ame60] American Standards Association. American standard acoustical terminology, 1960. Definition 12.9 – Timbre.

[AN95] Agnar Aamodta and Mads Nygardb. Different roles and mutual dependencies of data, information, and knowledge an AI perspective on their integration. *Data & Knowledge Engineering*, 16(3):191–222, September 1995.

[AP02] J.-J. Aucouturier and F Pachet. Finding songs that sound the same. In *Proceedings of IEEE Benelux Workshop on Model based Processing and Coding of Audio*. University of Leuven, Belgium, November 2002. Invited Talk.

[AR06] Samer Abdallah and Yves Raimond. Proposal for a common multimedia ontology framework: Information management for music analysis systems. AceMedia (`http://www.acemedia.org/`) call for proposals on a common multimedia ontology framework., April 2006.

[ARS06] Samer Abdallah, Yves Raimond, and Mark Sandler. An ontology-based approach to information management for music analysis systems. In *Proceedings of 120th AES convention*, 2006.

[AS06] Danny Ayers and Henry Story. AtomOwl vocabulary specification. Working draft, June 2006. `http://bblfish.net/work/atom-owl/2006-06-06/`. Last accessed June 2007.

[aud] Audioscrobbler: The social music technology playground. Web site. `http://audioscrobbler.net/`. Last accessed April 2008.

[BADW04] Christopher Brewster, Harith Alani, Srinandan Dasmahapatra, and Yorick Wilks. Data driven ontology evaluation. In *Proceedings of the International Conference on Language Resources and Evaluation*, pages 164–168, Lisbon, Portugal, 2004.

[Bal96] Mira Balaban. The music structures approach to knowledge representation for music processing. *Computer Music Journal*, 20(2):96–111, Summer 1996.

[Bau05] S. Baumann. A music library in the palm of your hand. In *Proceedings of the Contactforum on Digital libraries for musical audio (Perspectives and tendencies in digitalization, conservation, management and accessibility)*, Brussels, June 2005.

[BB03] Alex Borgida and Ronald J. Brachman. *The Description Logic handbook: theory, implementation, and applications*, chapter Conceptual modeling with description logics, pages 349–372. Cambridge University Press New York, NY, USA, 2003. ISBN:0-521-78176-0.

[BB08] Christian Becker and Christian Bizer. DBpedia mobile: A location-enabled linked data browser. In *Proceedings of the Linked Data on the Web workshop, colocated with the World-Wide-Web Conference*, Beijing, China, April 2008.

[BBF08] Uldis Bojars, John G. Breslin, and Aidan Finn. Using the Semantic Web for linking and reusing data across Web 2.0 communities. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(1), 2008.

[BBP$^+$08] Uldis Bojars, John Breslin, Vassilios Peristeras, Giovanni Tummarello, and Stefan Decker. Interlinking the social web with semantics. *Intelligent Systems, IEEE*, 23(3):29–40, May-June 2008.

[BC06] Christian Bizer and Richard Cyganiak. D2R server publishing relational databases on the Semantic Web. In *Proceedings of the 5th International Semantic Web conference*, 2006.

[BC08] Christian Bizer and Richard Cyganiak. Quality-driven information filtering using the WIQA policy framework. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2008.

[BCD03] D. Bainbridge, S. J. Cunningham, and S. J. Downie. How people describe their music information needs: A grounded theory analysis of music queries. In *Proceedings of the 4th International Conference on Music Information Retrieval*, 2003.

[BCG$^+$05] Stefano Beco, Barbara Cantalupo, Ludovico Giammarino, Nikolaos Matskanis, and Mike Surridge. OWL-WS: A workflow ontology for dynamic grid service composition. In *Proceedings of the First International Conference on e-Science and Grid Computing*, pages 148 – 155. IEEE Computer Society Washington, DC, USA, 2005.

[BCM$^+$03] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Application.* Cambridge University Press, Cambridge, UK, 2003. ISBN 0-521-78176-0.

[BDDS04] J.P. Bello, C. Duxbury, M.E. Davies, and . Sandler. On the use of phase and energy for musical onset detection in the complex domain. *IEEE Signal Processing Letter*, 11(6):553–556, June 2004.

[Bec05] Ralph Becket. *Mercury tutorial*, 2005.

[BG04] Dan Brickley and R.V. Guha. RDF vocabulary description language 1.0: RDF schema. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-schema/`. Last accessed September 2008.

[BHAR07] Chris Bizer, Tom Heath, Danny Ayers, and Yves Raimond. Interlinking open data on the web. In *Demonstrations Track, 4th European Semantic Web Conference, Innsbruck, Austria*, 2007.

[BHS05] Franz Baader, Ian Horrocks, and Ulrike Sattler. *Description Logics as Ontology Languages for the Semantic Web*, volume 2605/2005 of *Lecture Notes in Computer Science*, pages 228–248. Springer Berlin / Heidelberg, 2005.

[Biz07] Chris Bizer. *Quality-Driven Information Filtering in the Context of Web-Based Information Systems.* PhD thesis, Freie Universitt Berlin, 2007.

[BK93] Anthony J. Bonner and Michael Kifer. Transaction logic programming. In *Proceedings of the International Conference on Logic Programming*, pages 257–279, 1993.

[BK94] Anthony J. Bonner and Michael Kifer. An overview of transaction logic. *Theoretical Computer Science*, 133:205–265, 1994.

[BK95] Anthony J. Bonner and Michael Kifer. Transaction logic programming (or, a logic of procedural and declarative knowledge). Technical Report Tech. Rep. CSRI-323, Computer Systems Research Institute, University of Toronto, November 1995.

[BK96] Anthony Bonner and Michael Kifer. Concurrency and communication in transaction logic. In *Proceedings of the Joint International Conference and Symposium on Logic Programming*, 1996.

[BKN02] Stephan Baumann, Andreas Klter, and Marie Norlien. Using natural language input and audio analysis for a human-oriented mir system. In *Proceedings of Web Delivery of Music (WEDELMUSIC)*, 2002.

[BKWW99] T. Blum, D. Keislar, J. Wheaton, and E. Wold. Method and article of manufacture for content-based analysis, storage, retrieval and segmentation of audio information. U.S. Patent 5,918,223, June 1999.

[BL89] Tim Berners-Lee. Information management: A proposal. Proposal, circulated for comments at CERN, March 1989. Available at `http://www.w3.org/History/1989/proposal.html`. Last accessed September 2008.

[BL06a] Tim Berners-Lee. Linked data. World wide web design issues, July 2006. Available at `http://www.w3.org/DesignIssues/LinkedData.html`. Last accessed September 2007.

[BL06b] Tim Berners-Lee. Notation 3 (N3) - a readable RDF syntax. World Wide Web Design Issue, March 2006. `http://www.w3.org/DesignIssues/Notation3.html`. Last accessed August 2008.

[BL07] David M. Blei and John D. Lafferty. A correlated topic model of *Science*. *The Annals of Applied Statistics*, 1(1):17–35, 2007.

[BLCC$^+$06] Tim Berners-Lee, Yuhsin Chen, Lydia Chilton, Dan Connolly, Ruth Dhanaraj, James Hollenbach, Adam Lerer, and David Sheets. Tabulator: Exploring and analyzing linked data on the semantic web. In *Proceedings of the 3rd International Semantic Web User Interaction Workshop*, 2006.

[BLCK$^+$08] Tim Berners-Lee, Dan Connolly, Lalana Kagal, Yosi Scharf, and Jim Hendler. N3Logic : A logical framework for the world wide web. *Theory and Practice of Logic Programming*, 8(3):249–269, May 2008.

[BLHL01] Tim Berners-Lee, J. Hendler, and O. Lassila. The semantic web. *Scientific American*, pages 34–43, May 2001.

[BM07] D. Brickley and L. Miller. FOAF vocabulary specification. Working draft, May 2007. Available at `http://xmlns.com/foaf/0.1/`. Last accessed September 2007.

[BMS95] A. Brink, S. Marcus, and V.S. Subrahmanian. Heterogeneous multimedia reasoning. *Computer, IEEE*, 28(9):33–39, September 1995.

[BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent Dirichlet Allocation. *The Journal of Machine Learning Research*, 3(3):993–1022, March 2003.

[Boh99] Philip V. Bohlman. *Rethinking Music*, chapter Ontologies of Music, pages 17–34. Oxford University Press, 1999.

[Bol07] Susanne Boll. Share it, reveal it, reuse it, and push multimedia into a new decade. *IEEE Multimedia*, 14(4):14–19, 2007.

[bpe] Business process execution language for web services version 1.1. Specification. `http://www-128.ibm.com/developerworks/library/specification/ws-bpel/`. Last accessed June 2008.

[BRBM78] William Buxton, William Reeves, Ronald Baecker, and Leslie Mezei. The use of hierarchy and instance in a data structure for computer music. *Computer Music Journal*, 2(4):10–20, 1978.

[Bro91] J.C. Brown. Calculation of a Constant-Q spectral transform. *The Journal of the Acoustical Society of America*, 89(1):425–434, January 1991.

[BS97] Marko Balabanovi and Yoav Shoham. Fab: content-based, collaborative recommendation. *Communications of the ACM*, 40(3):66–72, March 1997.

[BS04] Christian Bizer and Andy Seaborne. D2RQ treating non-RDF databases as virtual RDF graphs. In *Proceedings of the 3rd International Semantic Web Conference*, 2004.

[BSR96] A.J. Bonner, A. Shrufi, and S. Rozen. Database requirements for workflow management in a high-throughput genome laboratory. In *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems*, Athens, Georgia, May 1996.

[BT05] Stuart Bray and George Tzanetakis. Distributed audio feature extraction for music. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[Bus45] Vannevar Bush. As we may think. *Atlantic Monthly*, July 1945. Available online at `http://www.theatlantic.com/doc/194507/bush`. Last accessed 4th of January, 2008.

[BW01] M.A. Bartsch and G.H. Wakefield. To catch a chorus: using chroma-based representations for audio thumbnailing. In *IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics*, pages 15–18, New Platz, NY, USA, 2001. ISBN: 0-7803-7126-7.

[Can] Chris Cannam. Vamp plugins. Web site. `http://www.vamp-plugins.org/`. Last accessed June 2008.

[CBHS05a] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs. *Journal of Web Semantics*, 2005.

[CBHS05b] Jeremy J. Carroll, Christian Bizer, Pat Hayes, and Patrick Stickler. Named graphs, provenance and trust. In *Proceedings of the 14th international conference on World Wide Web*, volume Semantic Web foundations, pages 613–622, Chiba, Japan, 2005. ACM Press.

[CBKH05] P. Cano, E. Batlle, T. Kalker, and J. Haitsma. A review of audio fingerprinting. *The Journal of VLSI Signal Processing*, 41(3):271 – 284, 2005.

[CC08] Oscar Celma and Pedro Cano. From hits to niches? or how popular artists can bias music recommendation and discovery. In *2nd Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition (ACM KDD)*, Las Vegas, USA, August 2008.

[CCIM95] Antonio Camurri, Alessandro Catorcini, Carlo Innocenti, and Alberto Massari. Music and multimedia knowledge representation and reasoning: The harp system. *Computer Music Journal*, 19(2):34–58, Summer 1995.

[CCPP95] F. Casati, S. Ceri, B Pernici, and G. Pozzi. *OOER '95: Object-Oriented and Entity-Relationship Modeling*, volume 1021/1995 of *Lecture Notes in Computer Science*, chapter Conceptual modeling of workflows, pages 341–354. Springer Berlin / Heidelberg, 1995.

[CDH$^+$07] Oscar Celma, Stamatia Dasiopoulou, Michael Hausenblas, Suzanne Little, Chrisa Tsinaraki, and Raphaël Troncy. MPEG-7 and the Semantic Web. W3C Incubator Group Editor's Draft, August 2007. `http://www.w3.org/2005/Incubator/mmsem/XGR-mpeg7/`. Last accessed September 2008.

[CF00] William W. Cohen and Wei Fan. Web-collaborative filtering: recommending music by crawling the web. *Computer Networks*, 33(1-6):685–698, June 2000.

[CGVD08] Nik Corthaut, Sten Govaerts, Katrien Verbert, and Erik Duval. Connecting the dots: Music metadata generation, schemas and applications. In *Proceedings of the International Conference on Music Information Retrieval*, Philadelphia, PA, USA, September 2008.

[cho] Chord transcriptions for researchers. Web dataset. `http://chordtranscriptions.net/`. Last accessed July 2008.

[Chu36] Alonzo Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:345–363, 1936.

[CKF$^+$04] Pedro Cano, Markus Koppenberger, Sira Ferradans, Alvaro Martinez, Fabien Gouyon, Vegard Sandvold, Vadim Tarasov, and Nicolas Wack. MTG-DB: A repository for music audio processing. In *Proceedings of the Fourth International Conference on Web Delivering of Music*, pages 2–9, Sept. 2004.

[Cla] Kendall Clark. SPARQL service advertisement and discovery language (SADDLE). Working draft. `http://www.w3.org/2001/sw/DataAccess/proto-wd/saddle.html`. Last accessed July 2008.

[CLSB06] Chris Cannam, Christian Landone, Mark Sandler, and Juan Pablo Bello. The Sonic Visualiser: A visualisation platform for semantic descriptors from musical signals. In *Proceedings of the International Conference on Music Information Retrieval*, 2006.

[CM84] W.F. Clocksin and C.S. Mellish. *Programming in Prolog*. Springer-Verlag New York, Inc., New York, NY, USA, 1984.

[CM04] Ben Caplan and Carl Matheson. Can a musical work be created? *British Journal of Aesthetics*, 44(2):113–134, April 2004.

[Cod70] E. F. Codd. A relational model of data for large shared data banks. *Communications of the ACM*, 13(6):377–387, 1970.

[Con07] Nepomuk Consortium. Nepomuk - the social semantic desktop. Web site, 2007. `http://nepomuk.semanticdesktop.org/`. Last accessed August 2008.

[CR08] Oscar Celma and Yves Raimond. Zempod: A semantic web approach to podcasting. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(2):162–169, March 2008.

[CRH05] O. Celma, M. Ramirez, and P. Herrera. Foafing the music: A music recommendation system based on rss feeds and user preferences. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[Dam64] Fred J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, March 1964.

[Dan93] Roger B. Dannenberg. Music representation issues, techniques, and systems. *Computer Music Journal*, 17(3):20–30, Autumn 1993.

[DCG$^+$08] John Domingue, Liliana Cabral, Stefania Galizia, Vlad Tanasescu, Alessio Gugliotta, Barry Norton, and Carlos Pedrinaci. Irs-iii: A broker-based approach to semantic web services. *Web Semantics: Science, Services and Agents on the World Wide Web*, 6(2):109–132, April 2008.

[DET05] J. Stephen Downie, Andreas F. Ehmann, and David Tcheng. Music-to-knowledge (M2K): a prototyping and evaluation environment for music information retrieval research. In *SIGIR '05: Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 676–676, New York, NY, USA, 2005. ACM Press.

[DF06] Li Ding and Tim Finin. Characterizing the semantic web on the web. In *Proceedings of the 5th International Semantic Web Conference*, volume 4273/2006 of *Lecture Notes in Computer Science*, pages 242–257, Athens, USA, November 2006. Springer Berlin / Heidelberg.

[DH93] Peter Desain and Henkjan Honing. Tempo curves considered harmful. *Contemporary Music Review*, 7(2):123–138, 1993.

[DJ05] Ian Davis and Eric Vitiello Jr. Relationship: A vocabulary for describing relationships between people. Working draft, 2005.

[DKRR98] Hasan Davulcu, Michael Kifer, C. R. Ramakrishnan, and I. V. Ramakrishnan. Logic based modeling and analysis of workflows. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 25–33, New York, NY, USA, 1998. ACM Press.

[DN05] Ian Davis and Richard Newman. Expression of core FRBR concepts in RDF. Working draft, 2005. Available at `http://vocab.org/frbr/core`. Last accessed September 2007.

[Dod00] Julian Dodd. Musical works as eternal types. *The British Journal of Aesthetics*, 40(4):424–440, 2000.

[Dod04] Leigh Dodds. Musicbrainz metadata vocabulary. Online RDF Vocabulary, February 2004. `http://www.ldodds.com/projects/musicbrainz/schema/mb.html`. Last accessed September 2008.

[Doe03] Martin Doerr. The CIDOC conceptual reference module: an ontological approach to semantic interoperability of metadata. *AI Magazine*, 24(3):75–92, September 2003. The CIDOC conceptual reference model is available at `http://cidoc.ics.forth.gr/`. Last accessed February 2008.

[Dow06] J. Stephen Downie. The music information retrieval evaluation exchange (MIREX). *D-Lib Magazine*, 12(12), December 2006. `http://dlib.org/dlib/december06/downie/12downie.html`. Last accessed July 2008.

[DP07] Matthew E.P. Davies and Mark D. Plumbley. Context-dependent beat tracking of musical audio. *IEEE Transactions on Audio, Speech, and Language Processing*, 15(3):1009–1020, March 2007.

[dRGS07] Dave de Roure, Carole Goble, and R. Stevens. Designing the myExperiment virtual research environment for the social sharing of workflows. In *IEEE International Conference on e-Science and Grid Computing*, pages 603–610, Bangalore, 2007.

[dSMF06] Paulo Pinheiro da Silva, Deborah L. McGuinness, and Richard Fikes. A proof markup language for semantic web services. *Information Systems*, 31(4-5):381–395, June-July 2006.

[duba] DCMI namespace for the Dublin Core Metadata Element Set, version 1.1. RDF vocabulary. `http://purl.org/dc/elements/1.1/`. Last accessed February 2008.

[dubb] The Dublin Core Metadata Initiative. Online. `http://dublincore.org/`. Last accessed February 2008.

[DZFJ05] Li Ding, Lina Zhou, Tim Finin, and Anupam Joshi. How the Semantic Web is being used: An analysis of FOAF documents. In *Proceedings of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS '05.*, pages 113c– 113c, January 2005.

[eas] Enabling access to sound archives through enrichment and retrieval. European Project Web-site. `http://www.easaier.org/`. Last accessed March 2008.

[ech] The echonest analyze API. Web service. `http://the.echonest.com/analyze/`. Last accessed June 2008.

[EDJ07] Andreas F. Ehmann, J. Stephen Downie, and M. Cameron Jones. The music information retrieval evaluation exchange "Do-It-Yourself" web service. In *Proceedings of the International Conference on Music Information Retrieval*, 2007.

[EE68] D. Engelbart and W. English. A research center for augmenting human intellect. *Reprinted in ACM SIGGRAPH Video Review 106 (1994), video made in 1968*, 1968.

[Ego06] Sergei Egorov. RDF graph patterns and templates. RDF vocabulary, 2006. `http://vocab.org/riro/gpt.html`. Last accessed July 2008.

[ES07] Jérome Euzenat and Pavel Shvaiko. *Ontology Matching*. Springer-Verlag New York, Inc. Secaucus, NJ, USA, 2007. ISBN: 3540496114.

[FD06] Tim Finin and Li Ding. Search engines for semantic web knowledge. In *Proceedings of XTech: Building Web 2.0*, Amsterdam, May 2006.

[FNH$^+$05] A.R.J. Francois, R. Nevatia, J. Hobbs, R.C. Bolles, and J.R. Smith. VERL: an ontology framework for representing and annotating video events. *IEEE Multimedia*, 12(4):76–86, October-December 2005.

[FPV98] A. Fuggetta, G.P. Picco, and G. Vigna. Understanding code mobility. *IEEE Transactions on Software Engineering*, 24(5):342–361, May 1998.

[FRS08] Gyorgy Fazekas, Yves Raimond, and Mark Sandler. A framework for producing rich musical metadata in creative music production. In *Proceedings of the 125th Audio Engineering Society convention*, 2008.

[FS69] Ivan P. Fellegi and Alan B. Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, Dec. 1969.

[fus03] FUSION project, 2003. Available at `http://metadata.net/sunago/fusion.htm`. Last accessed April 2008.

[G2̈9] Kurt Gödel. *Über die Vollständigkeit des Logikkalküls.* PhD thesis, University Of Vienna., 1929.

[Gal91] Antony Galton. Reified temporal theories and how to unreify them. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI'91)*, pages 1177–1182, Sydney, Australia, August 1991. ISBN 1-55860-160-0.

[Gar05] Roberto García. *A Semantic Web Approach to Digital Rights Management.* PhD thesis, Department of Technologies, Universitat Pompeu Fabra, Barcelona, Spain, 2005.

[GC05] Roberto García and Óscar Celma. Semantic integration and retrieval of multimedia metadata. In *Proceedings of the 5th International Workshop on Knowledge Markup and Semantic Annotation*, 2005.

[GHVD03] Benjamin N. Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description logic programs: Combining logic programs with description logic. In *Proceedings of the World Wide Web Conference.* ACM, 2003.

[GM07] Hugh Glaser and Ian C. Millard. RKB explorer: Application and infrastructure. In *Semantic Web Challenge, co-located with the International Semantic Web Conference*, Busan, Korea, November 2007.

[GMJ+07] H. Glaser, I. Millard, A. Jaffri, T. Lewy, and B. Dowling. On coreference and the semantic web. In *Proceedings of the International Semantic Web Conference*, Karlsruhe, Germany, October 2007.

[GO94] Thomas R. Gruber and Gregory R. Olsen. An ontology for engineering mathematics. In Jon Doyle, Piero Torasso, and Erik Sandewall, editors, *Fourth International Conference on Principles of Knowledge Representation and Reasoning*, Gustav Stresemann Institut, Bonn, Germany, 1994. `http://www-ksl.stanford.edu/knowledge-sharing/papers/engmath.html`. Last accessed Februart 2008.

[Gol91] Charles F. Goldfarb. HyTime: A standard for structured hypermedia interchange. *IEEE Computer Magazine*, 24(8):81–84, Aug. 1991.

[GPFLC04] Asunción Gómez-Pérez, Mariano Fernández-López, and Oscar Corcho. *Ontological Engineering: With Examples from the Areas of Knowledge Management, E-commerce and the Semantic Web.* Springer, 2004. ISBN 1852335513, 9781852335519.

[GR07] Frederick Giasson and Yves Raimond. Music ontology specification. Online ontology, February 2007. Available at `http://musicontology.com`. Last accessed September 2008.

[Gro94] William I. Grosky. Multimedia information systems. *Multimedia, IEEE*, 1:12–24, Spring 1994.

[Gro97] William I. Grosky. Managing multimedia information in database systems. *Commun. ACM*, 40(12):72–80, 1997.

[Gru93] T.R. Gruber. A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199–220, 1993. Online definition at `http://www-ksl.stanford.edu/kst/what-is-an-ontology.html`. Last accessed September 2008.

[GS04] T. Griffiths and M. Steyvers. Finding scientific topics. In *Proceedings of the National Academy of Sciences*, 2004.

[GW02] Nicola Guarino and Christopher Welty. Evaluating ontological decisions with ONTOCLEAN. *Communications of the ACM*, 45(2):61–65, 2002.

[Hay96] Patrick Hayes. A catalog of temporal theories. Technical Report UIUC-BI-AI-96-01, University of Illinois, 1996.

[Hay03] C. Hayes. *Smart Radio: Building community-Based Internet Music Radio.* PhD thesis, Trinity College Dublin, October 2003.

[Hay04] Patrick Hayes. RDF semantics. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-mt/`. Last accessed September 2008.

[HBW$^+$05] P. Herrera, J. Bello, G. Widmer, M. Sandler, O. Celma, F. Vignoli, E. Pampalk, P. Cano, S. Pauws, and X. Serra. SIMAC: Semantic interaction with music audio contents. In *Proceedings of the 2nd European Workshop on the Integration of Knowledge, Semantic and Digital Media Technologies. Savoy Place, London.*, 2005.

[Hef01] Jeff Heflin. *Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment.* PhD thesis, University of Maryland, College Park, 2001.

[Hei08] Gregor Heinrich. Parameter estimation for text analysis. Technical report, University of Leipzig & vsonix GmbH, Darmstadt, Germany, April 2008.

[Her07] Ivan Herman. Musicbrainz instrument taxonomy in SKOS. Working draft, 2007. `http://purl.org/ontology/mo/mit/`. Last accessed March 2008.

[HHR08] Michael Hausenblas, Wolfgang Halb, and Yves Raimond. Scripting user contributed interlinking. In *Proceedings of the Scripting for the Semantic Web workshop, colocated with the European Semantic Web Conference*, May 2008. `http://www.semanticscripting.org/SFSW2008/papers/6.pdf`. Last accessed June 2008.

[HHRH08] M. Hausenblas, W. Halb, Y. Raimond, and T. Heath. What is the size of the Semantic Web? In *International Conference on Semantic Systems (I-SEMANTICS)*, 2008.

[HL05a] Goffredo Haus and Maurizio Longari. A multi-layered, timebased music description approach based on XML. *Computer Music Journal*, 29(1):70 – 85, February 2005.

[HL05b] Jane Hunter and Suzanne Little. A framework to enable the semantic inferencing and querying of multimedia content. *International Journal of Web Engineering and Technology (IJWET)*, 2, December 2005.

[HM07] Tom Heath and Enrico Motta. Revyu.com: A reviewing and rating site for the web of data. In *The Semantic Web, Proceedings of the International Semantic Web conference*, volume 4825/2008 of *Lecture Notes in Computer Science*, pages 895–902. Springer Berlin / Heidelberg, 2007.

[HMP$^+$04] Alan Hevner, Salvatore March, Jinsoo Park, , and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28:75–105, 2004.

[Hod93] Wilfrid Hodges. *Model Theory*. Cambridge University Press, 1993. ISBN 0-521-30442-3.

[Hon01] Henkjan Honing. From time to time: The representation of timing and tempo. *Computer Music Journal*, 25(3):50–61, Autumn 2001.

[HP04] Jerry R. Hobbs and Feng Pan. An ontology of time for the Semantic Web. *ACM Transactions on Asian Language Processing (TALIP): Special issue on Temporal Information Processing*, 3(1):66–85, March 2004.

[HPSB$^+$04] Ian Horrocks, Peter F. Patel-Schneider, Harold Boley, Said Tabet, Benjamin Grosof, and Mike Dean. SWRL: A semantic web rule language combining OWL and RuleML. W3C Member Submission, May 2004. `http://www.w3.org/Submission/2004/SUBM-SWRL-20040521/`. Last accessed March 2008.

[HRH08a] Wolfgang Halb, Yves Raimond, and Michael Hausenblas. Building linked data for both humans and machines - how to get statistical data about 500 million europeans onto the Semantic Web. In *Proceedings of the Linking Data on the Web workshop, co-located with the World Wide Web conference*, 2008.

[HRH08b] Wolfgang Halb, Yves Raimond, and Michael Hausenblas. RDFizing and interlinking the eurostat data set effort. Online dataset, January 2008. `http://riese.joanneum.at/`. Last accessed June 2008.

[HS05] Christopher Harte and Mark Sandler. Automatic chord identifcation using a quantised chromagram. In *Proceedings of the 118th Audio Engineering Society Convention*, May 2005.

[HS08] Bernhard Haslhofer and Bernhard Schandl. The OAI2LOD server: Exposing OAI-PMH metadata as linked data. In *Proceedings of the first Linked Data on the Web workshop, co-located with the World Wide Web conference*, Beijing, China, April 2008. Available at `http://events.linkeddata.org/ldow2008/papers/03-haslhofer-schandl-oai2lod-server.pdf`. Last accessed July 2008.

[HSAG05] Christopher Harte, Mark Sandler, Samer Abdallah, and Emilia Gomez. Symbolic representation of musical chords: A proposed syntax for text annotations. In *Proceedings of the International Conference on Music Information Retrieval*, 2005.

[HSW91] Mitch Harris, Alan Smaill, and Geraint Wiggins. Representing music symbolically. In *Proceedings of IX Colloquium on Musical Informatics, Associazione di Informatica Musicale Italiana*, pages 55–69, Venice, 1991.

[Hun96] Samuel Y. K. Hung. Implementation and performance of transaction logic in prolog. Master's thesis, Graduate Department of Computer Science, University of Toronto, 1996. Available at `ftp://ftp.cs.toronto.edu/pub/bonner/papers/transaction.logic/theses/hung.ps`. Last accessed September 2007.

[Hun03] Jane Hunter. Enhancing the semantic interoperability of multimedia through a core ontology. *IEEE Transactions on Circuits and Systems for Video Technology*, 13:49–58, Jan 2003.

[HvOH06] Michiel Hildebrand, Jacco van Ossenbruggen, and Lynda Hardman. *The Semantic Web - ISWC 2006*, volume 4273/2006 of *Lecture Notes in Computer Science*, chapter /facet: A Browser for Heterogeneous Semantic Web Repositories, pages 272–285. Springer Berlin / Heidelberg, 2006.

[id3] ID3 audio file data tagging format. Web site. `http://www.id3.org/`. Last accessed March 2008.

[Jaf85] David Jaffe. Ensemble timing in computer music. *Computer Music Journal*, 9(4):38–48, Winter 1985.

[Jar89] M. A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. *Journal of the American Statistical Association*, 89:414–420, 1989.

[JGM07] Afraz Jaffri, Hugh Glaser, and Ian Millard. *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops*, volume 4806/2007 of *Lecture Notes in Computer Science*, chapter URI Identity Management for Semantic Web Data Integration and Linkage, pages 1125–1134. Springer Berlin / Heidelberg, 2007.

[JGPP95] N. Juristo, A. Gomez-Perez, and J. Pazos. *Towards Very Large Knowledge Bases*, chapter Evaluation and assessment of knowledge sharing technology, pages 289–296. IOS Press, 1995.

[JW04] Ian Jacobs and Norman Walsh. Architecture of the World Wide Web, volume one. W3C Recommendation, December 2004. `http://www.w3.org/TR/webarch/`. Last accessed July 2008.

[Kan07] Masahide Kanzaki. Music vocabulary. Web ontology, March 2007. `http://www.kanzaki.com/ns/music`. Last accessed September 2008.

[KC04] Graham Klyne and Jeremy J. Carroll. Resource description framework (RDF): Concepts and abstract syntax. W3C Recommendation, February 2004. `http://www.w3.org/TR/rdf-concepts/`. Last accessed September 2008.

[Kie03] Bartosz Kiepuszewski. *Expressiveness and Suitability of Languages for Control Flow Modelling in Workflows.* PhD thesis, Faculty of Information Technology, Queensland University of Technology, February 2003.

[KLW95] Michael Kifer, Georg Lausen, and James Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM*, 15(4):741–843, July 1995.

[Kow79] Robert Kowalski. *Logic for Problem Solving.* North Holland Publishing Co., 1979.

[KS86] Robert Kowalski and Marek Sergot. A logic-based calculus of events. *New Generation Computing*, 4:67–95, 1986.

[KVV06] Markus Krtzsch, Denny Vrandei, and Max Vlkel. Semantic mediawiki. In *The Semantic Web - ISWC 2006, Proceedings of the International Semantic Web conference*, volume 4273/2006 of *Lecture Notes in Computer Science*, pages 935–942. Springer Berlin / Heidelberg, November 2006.

[LAB$^+$06] Bertram Ludscher, Ilkay Altintas, Chad Berkley, Dan Higgins, Efrat Jaeger, Matthew Jones, Edward A. Lee, Jing Tao, and Yang Zhao. Scientific workflow management and the Kepler system. *Concurrency and Computation: Practice & Experience*, 18(10):1039 – 1065, August 2006.

[LDJ07] Jin Ha Lee, J. Stephen Downie, and M. Cameron Jones. Preliminary analyses of information features provided by users for identifying music. In *Proceedings of the International Conference on Music Information Retrieval*, 2007.

[Lev66] V. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, Feb. 1966.

[Lev80] Jerrold Levinson. What a musical work is. *The Journal of Philosophy*, 77(1):5–28, January 1980.

[LH97] Sean Luke and James Hendler. Web agents that work. *Multimedia, IEEE*, 4(3):76–80, Jul.-Sep. 1997.

[LH01] Carl Lagoze and Jane Hunter. The ABC ontology and model. *Journal of Digital Information*, 2(2), 2001.

[LS08] Mark Levy and Mark Sandler. Structural segmentation of musical audio by constrained clustering. *IEEE Transactions on Audio, Speech, and Language Processing*, 16(2):318–326, Feb. 2008.

[LTGP04] Adolfo Lozano-Tello and Asuncion Gomez-Perez. ONTOMETRIC: A method to choose the appropriate ontology. *Journal of Database Management*, 15(2):1–18, 2004.

[LWB08] Andreas Langegger, Wolfram W, and Martin Blchl. A Semantic Web middleware for virtual data integration on the web. In *The Semantic Web: Research and Applications. Proceedings of the European Semantic Web Conference*, Lecture Notes in Computer Science, pages 493–507. Springer Berlin / Heidelberg, 2008.

[m3f] M3F meta format. Working draft. `http://wiki.xiph.org/index.php/M3F`. Last accessed March 2008.

[McD82] Drew McDermott. A temporal logic for reasoning about processes and plans. *Cognitive Science: A Multidisciplinary Journal*, 6(2):101–155, 1982.

[MdS04] Deborah L. McGuinness and Paulo Pinheiro da Silva. Explaining answers from the semantic web: The inference web approach. *Journal of Web Semantics*, 1(4):397–413, October 2004.

[ME05] Michael Mandel and Dan Ellis. Song-level features and support vector machines for music classification. In *Proceedings of the International Conference on Music Information Retrieval*, pages 594–599, London, 2005.

[MGMR02] S. Melnik, H. Garcia-Molina, and E. Rahm. Similarity flooding: a versatile graph matching algorithm and itsapplication to schema matching. In *Proceedings of the 18th International Conference on Data Engineering*, pages 117–128, San Jose, CA, USA, February-March 2002.

[MH69] John McCarthy and Patrick J. Hayes. *Some Philosophical Problems from the Standpoint of Artificial Intelligence.* 1969.

[MHV03] Pedro J. Moreno, Purdy P. Ho, and Nuno Vasconcelos. A Kullback-Leibler divergence based kernel for SVM classification in multimedia applications. In *Proceedings of Neural Information Processing Systems*, Vancouver, Canada, 2003.

[MM03] Daniel J. Mandell and Sheila A. McIlraith. Adapting BPEL4WS for the semantic web: The bottom-up approach to web service interoperation. In *The Semantic Web - International Semantic Web Conference*, volume 2870/2003 of *Lecture Notes in Computer Science*, pages 227–241. Springer Berlin / Heidelberg, 2003.

[MMF06] Daniel McEnnis, Cory McKay, and Ichiro Fujinaga. Overview of OMEN. In *Proceedings of the International Conference on Music Information Retrieval*, 2006.

[MMWB05] Alistair Miles, B. Matthews, M. Wilson, and D. Brickley. SKOS core: Simple knowledge organisation for the web. In *roceedings of the International Conference on Dublin Core and Metadata Applications (DC-2005),*, pages 5–13, Madrid, 2005.

[MNJ05] Prasenjit Mitra, Natalya F. Noy, and Anuj R. Jaiswal. Ontology mapping discovery with uncertainty. In *Proceedings of the 4th International Semantic Web Conference*, Lecture Notes in Computer Science, Galway, Ireland, November 6-10 2005. Springer.

[MPT07] Christian Morbidoni, Axel Polleres, and Giovanni Tummarello. Who the FOAF knows alice? a needed step toward Semantic Web pipes. In *Proceedings of the First International Workshop Workshop "New forms of reasoning for the Semantic Web: scalable, tolerant and dynamic", co-located with the International Semantic Web Conference and the Asian Semantic Web Conference*, Busan, Korea, November 2007.

[MS02] Alexander Maedche and Steffen Staab. *Knowledge Engineering and Knowledge Management: Ontologies and the Semantic Web*, volume 2473/2002 of *Lecture Notes in Computer Science*, chapter Measuring Similarity between Ontologies, pages 15–21. Springer Berlin / Heidelberg, 2002.

[MSSvE07] Antoni Mylka, Leo Saumermann, Michael Sintek, and Ludger van Elst. NEPOMUK ID3 ontology. Online ontology, May 2007. `http://www.semanticdesktop.org/ontologies/2007/05/10/nid3/`. Last accessed September 2008.

[msWRS06] m.c. schraefel, Max Wilson, Alistair Russell, and Daniel Smith. mSpace: improving information access to multimedia domains with multimodal exploratory search. *Communications of the ACM*, 49(4):47–49, April 2006.

[MSZ01] S.A. McIlraith, T.C. Son, and Honglei Zeng. Semantic Web services. *Intelligent Systems, IEEE*, 16:46–53, Mar-Apr 2001.

[MvH04] Deborah L. McGuinness and Frank van Harmelen. OWL web ontology language overview. W3C Recommendation, February 2004. `http://www.w3.org/TR/owl-features/`. Last accessed March 2008.

[NL99a] Franck Nack and A.T. Lindsay. Everything you wanted to know about MPEG-7, part 1. *Multimedia, IEEE*, 6(3):65–77, Jul.-Sep. 1999.

[NL99b] Franck Nack and A.T. Lindsay. Everything you wanted to know about MPEG-7, part 2. *IEEE Multimedia*, 6(4):64–73, Oct.-Dec. 1999.

[NS07] K. Noland and M. Sandler. Signal processing parameters for tonality estimation. In *Proceedings of AES 122nd Convention*, Vienna, 2007.

[OGA+06] Tom Oinn, Mark Greenwood, Matthew Addis, M. Nedim Alpdemir, Justin Ferris, Kevin Glover Carole Golble, Antoon Goderis, Duncan Hull, Darren Marvin, Peter Li, Phillip Lord, Matthew R. Pocock, Martin Senger, Robert Stevens, Anil Wipat, and Chris Wroe. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice & Experience, Workflow in Grid Systems*, 18(10):1067 – 1100, August 2006.

[OS99] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing.* Prentice Hall Signal Processing Series, 1999. ISBN: 0-13-754920-2.

[otFRfBR98] IFLA Study Group on the Functional Requirements for Bibliographic Records. Functional requirements for bibliographic records - final report. UBCIM Publications - New Series Vol 19, September 1998. `http://www.ifla.org/VII/s13/frbr/frbr1.htm`. Last accessed February 2008.

[Pac05] F. Pachet. *Knowledge Management and Musical Metadata.* Idea Group, 2005.

[Pam06] Elias Pampalk. *Computational Models of Music Similarity and their Application in Music Information Retrieval.* PhD thesis, Vienna University of Technology, 2006.

[Pan07] Feng Pan. *Representing Complex Temporal Phenomena for the Semantic Web and Natural Language.* PhD thesis, Faculty of the Graduate School, University of Southern California, December 2007.

[PBAB05] Francois Pachet, Amaury La Burthe, Jean-Julien Aucouturier, and Anthony Beurive. Editorial metadata in electronic music distribution systems: Between universalism and isolationism. *Journal of New Music Research*, 34(2):173 184, 2005.

[PM04] Robert Porzel and Rainer Malaka. A task-based approach for ontology evaluation. In *Proceedings of the ECAI Workshop on Ontology Learning and Population*, 2004.

[Pol07] Axel Polleres. From SPARQL to rules (and back). In *Proceedings of the 16th international conference on World Wide Web*, 2007.

[Por80] M.F. Porter. An algorithm for suffix stripping. *Program*, 14(3):130137, 1980.

[PR07] Francois Pachet and Pierre Roy. Exploring billions of audio features. In *International Workshop on Content-Based Multimedia Indexing*, pages 227–235, Bordeaux, France, June 2007.

[PR08] Alexandre Passant and Yves Raimond. Combining social music and semantic web for music-related recommender systems. In *Proceedings of the Social Data on the Web workshop*, Karlsruhe, Germany, October 2008.

[PT05] C. Panagiotakis and G. Tziritas. A speech/music discriminator based on RMS and zero-crossings. *IEEE Transactions on Multimedia*, 7(1):155–166, Feb. 2005.

[RA06a] Yves Raimond and Samer A. Abdallah. The Event Ontology. OWL-DL ontology, 2006. Available at `http://purl.org/NET/c4dm/event.owl`. Last accessed September 2007.

[RA06b] Yves Raimond and Samer A. Abdallah. The Timeline Ontology. OWL-DL ontology, 2006. Available at `http://purl.org/NET/c4dm/timeline.owl`. Last accessed September 2007.

[Rai] Yves Raimond. DBTune. Online music-related datasets. Available at `http://dbtune.org/`. Last accessed June 2008.

[Rai07a] Yves Raimond. The audio features ontology. Working draft, February 2007. `http://purl.org/ontology/af/`. Last accessed March 2007.

[Rai07b] Yves Raimond. Audioscrobbler RDF service on DBTune. Online dataset, December 2007. `http://dbtune.org/last-fm/`. Last accessed June 2008.

[Rai07c] Yves Raimond. BBC John Peel Sessions in RDF. Online dataset, July 2007. Available at `http://dbtune.org/bbc/peel/`. Last accessed September 2007.

[Rai07d] Yves Raimond. DBTune semantic web publication of the jamendo dataset. Online dataset, April 2007. `http://dbtune.org/jamendo/`. Last accessed June 2008.

[Rai07e] Yves Raimond. DBTune semantic web publication of the magnatune dataset. Online dataset, March 2007. `http://dbtune.org/magnatune/`. Last accessed June 2008.

[Rai07f] Yves Raimond. Symbolic notation ontology. Working draft, December 2007. `http://purl.org/ontology/symbolic-music/`. Last accessed March 2008.

[Rai07g] Yves Raimond. Web identifiers for chords and associated chord ontology representation. Online dataset, October 2007. `http://motools.sourceforge.net/chord_draft_1/chord.html`symbolservice. Last accessed June 2008.

[Rai08a] Yves Raimond. BBC playcount linked data. Web dataset, June 2008. `http://dbtune.org/bbc/playcount/`. Last accessed July 2008.

[Rai08b] Yves Raimond. Musicbrainz linked data on the DBTune server. Online dataset, April 2008. `http://dbtune.org/musicbrainz/`. Last accessed June 2008.

[RASG07] Yves Raimond, Samer Abdallah, Mark Sandler, and Frederick Giasson. The music ontology. In *Proceedings of the International Conference on Music Information Retrieval*, pages 417–422, September 2007.

[Reg67] Eric Regener. Layered music-theoretic systems. *Perspectives of New Music*, 6(1):52–62, Autumn-Winter 1967.

[RH07] Tuukka RuotsaloTuukka Ruotsalo and Eero Hyvnen. An event-based approach for semantic metadata interoperability. In *Proceedings of the International Semantic Web Conference*, Busan, Korea, November 2007.

[RHCB07] Jenn Riley, Caitlin Hunter, Chris Colvard, and Alex Berry. Definition of a FRBR-based metadata model for the Indiana University Variations3 project. Technical report, Indiana University, 2007. `http://www.dlib.indiana.edu/projects/variations3/docs/v3FRBRreport.pdf`. Last accessed June 2008.

[RJ93] L. Rabiner and B. Juang. *Fundamentals of speech recognition*. Prentice-Hall, New Jersey, 1993.

[RK07] Dumitru Roman and Michael Kifer. Reasoning about the behavior of semantic web services with concurrent transaction logic. In *Proceedings of the 33rd international conference on Very large data base*, pages 627–638. VLDB Endowment, 2007.

[RKF08] Dumitru Roman, Michael Kifer, and Dieter Fensel. WSMO choreography: From abstract state machines to concurrent transaction logic. In *The Semantic Web: Research and Applications, Proceedings of the European Semantic Web Conference*, volume 5021/2008 of *Lecture Notes in Computer Science*, pages 659–673. Springer Berlin / Heidelberg, May 2008.

[RN95] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice-Hall, 1995. ISBN 0-13-360124-2.

[Rog05] P. Rogers. Podcasting and its role in semantic social networks, the web 2.0, and the semantic web. In N. Guarino and R. Poli, editors, *AIS SIGSEMIS Bulletin*, volume 2, pages 57–61, 2005.

[Rol02] Perry Roland. The music encoding initiative (MEI). In *Musical Applications using XML (MAX)*, 2002.

[RS08] Yves Raimond and Mark Sandler. A web of musical information. In *Proceedings of the International Conference on Music Information Retrieval*, Philadelphia, USA, 2008.

[RSS08] Yves Raimond, Christopher Sutton, and Mark Sandler. Automatic interlinking of music datasets on the semantic web. In *Proceedings of the Linked Data on the Web workshop, colocated with the World Wide Web Conference*, 2008.

[RSS09] Yves Raimond, Christopher Sutton, and Mark Sandler. Publishing and interlinking music-related data on the web. *Accepted for publication in IEEE Multimedia*, April-June 2009.

[Sad80] Stanley Sadie, editor. *The New GROVE Dictionary of Music and Musicians.* Macmillan Publishers Limited, London, 1980.

[SC08] Leo Sauermann and Richard Cyganiak. Cool uris for the semantic web. W3C Interest Group Note, March 2008. `http://www.w3.org/TR/cooluris/`. Last accessed June 2008.

[Sch04] Heinrich Schenker. *Der Tonwille: pamphlets in witness of the immutable laws of music: offered to a new generation of youth (2 volumes).* Oxford University Press, 2004. Trans. Ian Bent et al.

[Sch08] Markus Schedl. *Automatically Extracting, Analyzing, and Visualizing Information on Music Artists from the World Wide Web.* PhD thesis, Johannes Kepler Universitt Linz, Juni 2008.

[Sha95] R.A. Sharpe. Music, platonism and performance: Some ontological strains. *The British Journal of Aesthetics*, 35(1):38–48, 1995.

[Sha99] Murray P. Shanahan. The event calculus explained. In M. J. Woolridge and M. Veloso, editors, *Artificial Intelligence Today, Lecture Notes in AI no. 1600*, pages 409–430. Springer, 1999.

[Siz07] S. Sizov. What makes you think that? the Semantic Web's proof layer. *Intelligent Systems, IEEE*, 22(6):94–99, Nov.-Dec. 2007.

[SKW07] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago - a core of semantic knowledge. In *16th international World Wide Web conference*, 2007. Available at `http://www.mpi-inf.mpg.de/~suchanek/publications/www2007.pdf`. Last accessed january 2008.

[Sle00] Amalia F. Sleghel. An optimizing interpreter for concurrent transaction logic. Master's thesis, Department of Computer Science, University of Toronto, 2000. Available at `http://www.cs.toronto.edu/~bonner/ctr/download/Thesis_full.ps`. Last accessed September 2007.

[Slo97] D. Sloan. *Beyond MIDI: The Handbook of Musical Codes*, chapter HyTime and Standard Music Description Language: A Document-Description Approach, pages 469–490. MIT Press, 1997.

[SM83] Gerard Salton and Michael J. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill, Inc. New York, NY, USA, 1983.

[SOS+08] Ken Samuel, Leo Obrst, Suzette Stoutenberg, Karen Fox, Paul Franklin, Adrian Johnson, Ken Laskey, Deborah Nichols, Steve Lopez, and Jason Peterson. Translating OWL and Semantic Web rules into prolog: Moving toward description logic programs. *Theory and Practice of Logic Programming*, 8:301–322, 2008.

[Spo08] Manu Sporny. hAudio microformat 0.9. Draft Specification, May 2008. `http://microformats.org/wiki/haudio`. Last accessed June 2008.

[SR02] Matthew L. Saxton and John V. Richardson. *Understanding reference transactions*. Academic Press, May 2002.

[SRB+07] Francois Scharffe, Yves Raimond, Luc Barthelemy, Ying Ding, and Michael Luger. Publishing and accessing digital archives using the EASAIER framework. In *Proceedings of the First International Workshop on Cultural Heritage on the Semantic Web, co-located with the International Semantic Web Conference, Busan, Korea*, 2007.

[SRM07] Christopher Sutton, Yves Raimond, and Matthias Mauch. The OMRAS2 chord ontology. Working draft, October 2007. `http://purl.org/ontology/chord/`. Last accessed March 2008.

[SRSH08] Tom Scott, Yves Raimond, Patrick Sinclair, and Nicholas Humfrey. The Programmes Ontology. In *Proceedings of the XTech conference*, May 2008. Ontology available online at `http://www.bbc.co.uk/ontologies/programmes/`. Last accessed September 2008.

[SSST08] Bernhard Schueler, Sergej Sizov, Steffen Staab, and Duc Thanh Tran. Querying for meta knowledge. In *Proceeding of the 17th international conference on World Wide Web*, pages 625–634, Beijing, China, 2008. ACM New York, NY, USA.

[Sug07] Cassidy R. Sugimoto. Evaluating reference transactions in academic music libraries. Master's thesis, School of Information and Library Science of the University of North Carolina at Chapel Hill, 2007.

[SVN37] Stanley Smith Stevens, John Volkman, and Edwin B. Newman. A scale for the measurement of the psychological magnitude of pitch. *Journal of the Acoustical Society of America*, 8:185–190, 1937.

[SWH93] Alan Smaill, Geraint Wiggins, and Mitch Harris. *Computers and the Humanities*, volume 27, chapter Hierarchical music representation for composition and analysis, pages 7–17. Springer Netherlands, January 1993.

[Tau07] Joshua Tauberer. The 2000 U.S. Census: 1 billion RDF triples. Online dataset, August 2007. `http://www.rdfabout.com/demo/census/`. Last accessed June 2008.

[TC04] Raphaël Troncy and Jean Carrive. A reduced yet extensible audio-visual description language. In *DocEng '04: Proceedings of the 2004 ACM symposium on Document engineering*, pages 87–89, New York, NY, USA, 2004. ACM Press.

[Tro03] Raphal Troncy. Integrating structure and semantics into audio-visual documents. In *Proceedings of the International Semantic Web Conference*, volume LNCS 2870, pages 566–581, Sanibel Island, Florida, USA, October 2003.

[TS92] Jean Tague-Sutcliffe. The pragmatics of information retrieval experimentation, revisited. *Information Processing and Management, Special issue on evaluation issues in information retrieval*, 28(4):467–490, July-Aug. 1992.

[VDO03] Raphael Volz, Stefan Decker, and Daniel Oberle. Bubo - implementing OWL in rule-based. systems. In *Proceedings of the World Wide Web conference*, Budapest, Hungary, May 2003.

[Ver08] W.J.A. Verheijen. Efficient query processing in distributed rdf databases. Master's thesis, Eindhoven University of Technology, Department of Mathematics and Computer Science, February 2008. Available at `http://wwwis.win.tue.nl/~ksluijs/material/Verheijen-Master-Thesis-2008.pdf`. Last accessed June 2008.

[Vig04] F. Vignoli. Digital music interaction concepts: A user study. In *Proceedings of the 5th International Conference on Music Information Retrieval*, 2004.

[Vin04] Hugues Vinet. *Computer Music Modeling and Retrieval*, volume 2771/2004 of *Lecture Notes in Computer Science*, chapter The Representation Levels of Music Information, pages 193–209. Springer Berlin / Heidelberg, 2004.

[vONH05] Jacco van Ossenbruggen, Frank Nack, and Lynda Hardman. That obscure object of desire: Multimedia metadata on the web, part 2. *IEEE Multimedia*, 12(4):54–63, January–March 2005.

[VR96] Lluis Vila and Han Reichgelt. The token reification approach to temporal reasoning. *Artificial Intelligence*, 83(1):59–74, 1996.

[vR99] C.J. van Rijsbergen. *Information Retrieval, Second Edition*. Information Retrieval Group, Department of Computing Science, University of Galsgow, 1999. `http://www.dcs.gla.ac.uk/~iain/keith/index.htm`. Last accessed June 2008.

[VS07] Denny Vrandecic and York Sure. How to design better ontology metrics. In *Proceedings of the 4th European Semantic Web Conference*, volume 4519/2007 of *Lecture Notes in Computer Science*, pages 311–325. Springer Berlin / Heidelberg, 2007.

[WHM08] Jan Wielemaker, Zhisheng Huang, and Lourens Van Der Meij. SWI-Prolog and the Web. *Theory and Practice of Logic Programming*, 8(3):363–392, May 2008.

[Win99] W. Winkler. The state of record linkage and current research problems. Technical report, Statistical Research Division, U.S. Bureau of the Census, Washington, DC, 1999.

[YB04] Jia Yu and Rajkumar Buyya. A novel architecture for realizing grid workflow using tuple spaces. In *Proceedings of the fifth IEEE/ACM International Workshop on Grid Computing*, volume 119- 128, November 2004.

[YB05] Jia Yu and Rajkumar Buyya. A taxonomy of scientific workflow systems for grid computing. *ACM SIGMOD Record*, 34(3):44–49, September 2005.

[YGK$^+$06] Kazuyoshi Yoshii, Masataka Goto, Kazunori Komatani, Tetsuya Ogata, and Hiroshi G. Okuno. Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In *Proceedings of the International Conference on Music Information Retrieval*, 2006.

[YI99] A. Yoshitaka and T. Ichikawa. A survey on content-based retrieval for multimedia databases. *Knowledge and Data Engineering, IEEE Transactions on*, 11(1):81–93, Jan./Feb. 1999.

[ZF99] Eberhard Zwicker and Hugo Fastl. *Psychoacoustics, facts and models, 2nd edition.* Springer, Berlin, 1999.

[ZL08] Ying Zhang and Yuelin Li. A user-centered functional metadata evaluation of moving image collections. *Journal of the American Society for Information Science and Technology*, 59(8):1331–1346, March 2008.

[ZWG$^+$04] Jun Zhao, Chris Wroe, Carole Goble, Robert Stevens, Dennis Quan, and Mark Greenwood1. *The Semantic Web ISWC 2004*, volume 3298/2004, chapter Using Semantic Web Technologies for Representing e-Science Provenance, pages 92–106. Springer Berlin / Heidelberg, 2004.

# Part V

# Appendices

# Appendix A

# Musical audio analysis

Throughout this thesis, we use a number of example workflows deriving information from a digital audio signal. This process is similar to the information extraction process mentioned in § 2.2.2: we start from the representation (the discrete time signal, a series of values indexed by $\mathbb{Z}$ often resulting from the sampling of a continuous time signal) and extract information from it.

In the case of musical audio, one can automatically extract keys, chords, rhythm, timbre melody, etc [HBW+05]. All this extracted information is then usually used to drive a music information retrieval system. The results of an information extraction process for a large audio collection are stored in a database. The music information retrieval system then allows the user to search the database using this information. A simple example is a “query by example” music similarity system [Pam06], combining different facets of extracted information to return audio tracks that are similar to an input track.

In this section, we overview the different digital audio signal processing methods underlying the examples used throughout this thesis. Any argument about the accuracy of these methods is outside the scope of this thesis.

## A.1 Time domain and frequency domain

The discrete Fourier transform (DFT) of a discrete time signal $x(n)$ is defined as follows [OS99].

$$X(w) = \sum_n x(n) \cdot e^{-jwn} \tag{A-1}$$

where $w = 2\pi k/N$, $0 \leq k \leq N-1$ and $k$ is the frequency bin index.

The complex numbers $X(w)$ represent the amplitude and phase of the different sinusoidal components of the input audio signal. The DFT quantifies how much of a sinusoidal component is present in the signal. The result of this transform is called the frequency spectrum of the signal.

## A.2 STFT and spectrogram

The short-time Fourier transform (STFT) determines the spectrum of an audio signal for local sections (frames) of it. This transform can therefore be used to model the temporal evolution of the spectrum.

The STFT of a discrete time signal $x(n)$ is defined as follows.

$$X(m,w) = \sum_n x(n) \cdot W(n - mh) \cdot e^{-jwn} \tag{A-2}$$

where $m$ is the frame index, $h$ is the hop size and $W(n)$ is a window function centered around 0. Such window functions include:

- The Hamming window:

$$W(n) = 0.5386 - 0.46164 \cdot \cos\left(\frac{2\pi n}{N-1}\right)$$

- The Hann window:

$$W(n) = 0.5 \cdot \left(1 - \cos\left(\frac{2\pi n}{N-1}\right)\right)$$

where $0 \leq n \leq N-1$ and $N$ is the width of the window function.

The power spectrogram of an audio signal, identified by the `sig:spectrogram` predicate used in chapter 5, is then the power magnitude of the STFT.

$$S(m,w) = |X(m,w)|^2 \tag{A-3}$$

This representation is particularly interesting because the human auditory system applies a similar transformation. The *cochlea*, a part of the inner ear, holds primary auditory neurons which respond to different frequencies [ZF99]. However, although the spectrogram of a signal can give a good insights for e.g. identifying harmonic structures or transients, it still does not constitute meaningful information. We need to go a few steps further.

## A.3 Constant-Q and Chromagram

A transform related to the STFT is the Constant-Q [Bro91] transform. The frequency components of the DFT are separated by a constant frequency difference and have a constant resolution. The Constant-Q transform is similar to a DFT, but has a constant ratio of center frequency to resolution.

$$C(k) = \frac{1}{N(k)} \sum_{n=0}^{N(k)-1} W(k,n) \cdot x(n) \cdot e^{-\frac{j2\pi Qn}{N(k)}} \tag{A-4}$$

where $Q$ is a constant called the *quality factor*, defined as the ratio between the central frequency $f_k$ corresponding to the $k$-th bin and the filter width $\delta f_k$, $N(k) = Q\frac{f_s}{f_k}$ (where $f_s$ is the sampling frequency of the signal) is the window length corresponding to the $k$-th bin. The window function is now a function of $k$. For example, for the Hann window, it becomes:

$$W(k,n) = 0.5 \cdot (1 - \cos(\frac{2\pi n}{N(k)-1}))$$

This transform is particularly useful in that by using $f_k = f_0 * 2^{k/b}$ and an appropriate choice for the minimal central frequency $f_0$ and for the number of bins per octave $b$, the results of the transform directly correspond to musical notes. Another interesting feature of the Constant-Q transform is that it models a characteristic of the human auditory system by increasing the time resolution towards higher frequencies.

The chromagram (or Harmonic Pitch Class Profile) of an audio signal maps the Constant-Q frequency bins to a limited set of bins corresponding to the chromas in an octave, in the equal temperament chromatic scale.

$$G(k) = \sum_{m=0}^{M} |C(k + b \cdot m)| \qquad 1 \leq k \leq b \tag{A-5}$$

where $M$ is the number of octaves spanned by the Constant-Q spectrum, and $k$ is the chromagram bin number.

The chromagram of an audio signal can then be used for different harmony-related information extraction class. For example, it has been used for chord recognition [HS05], key estimation by matching it to key profiles [NS07] and for thumbnailing [BW01].

## A.4 Timbre

The American Standards Association defines the timbre as being "that attribute of sensation in terms of which a listener can judge that two sounds having the same loudness and pitch are dissimilar" [Ame60]. The timbre is the characteristic distinguishing, for example, several instruments playing the same note.

### A.4.1 Mel Frequency Cepstral Coefficients

There exists many different ways of computing a timbre model from an audio signal. The one we use as an example in § 5.2.4.3 is based on Mel Frequency Cepstral Coefficients (MFCCs [RJ93]). Intuitively, these coefficients estimate a *spectral envelope*, a curve in the frequency-magnitude space that envelopes the peaks of the short-time spectrum. The steps leading to such coefficients roughly model some of the characteristics of the human auditory system.

The cepstrum is the inverse Fourier transform of the log-spectrum, and is defined as follows.

$$c(n) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \log(X(w)) \cdot e^{jwn} \, dw \tag{A-6}$$

The Mel-scale [SVN37] is defined as follows.

$$m_{Mel} = 1127.01048 \cdot \log\left(1 + \frac{f}{700}\right) \tag{A-7}$$

where $f$ is the frequency in Hz.

A 1000 Hz tone corresponds to 1000 Mel. A tone with a pitch perceived twice as high is defined to have 2000 Mel. A tone perceived half as high is defined to have 500 Mel, etc. The Mel-cepstrum is the cepstrum computed after a frequency warping onto the Mel-scale. The $c(n)$ are then called MFCCs. The first coefficients correspond to the slowly changing spectral envelope and are the most useful as timbre descriptors [AP02].

The process identified by the `sig:mfcc` predicate in this thesis consists in cutting the audio signal in multiple frames, and deriving the first 20 MFCCs for each of these frames.

### A.4.2 Gaussian Modelling

Once we have a set of MFCCs describing timbral aspects of an audio signal, we derive a model out of it to reduce its dimensionality and variability. The example used in § 5.2.4.3 uses the approach described by Mandel and Ellis [ME05]. The MFCCs are modelled by a single Gaussian with full covariance. The process identified by the `sig:gaussian` predicate in this thesis consists of computing the mean and covariance of a set of observation vectors.

### A.4.3 Similarities amongst timbre models

The Gaussian modelling of the MFCCs extracted from the audio signal can serve as a basis for comparing signals among each other. A symmetrised version of the Kullback-Leibler divergence provides a relevant comparison of two Gaussians [MHV03]. The symmetrised Kullback-Leibler divergence between two Gaussian distributions $p$ and $q$ is given by:

$$KL(p||q) = Tr(\Sigma_p \cdot \Sigma_q^{-1}) + Tr(\Sigma_q \cdot \Sigma_p^{-1}) + Tr((\Sigma_p^{-1} + \Sigma_q^{-1}) \cdot (\mu_1 - \mu_2) \cdot (\mu_1 - \mu_2)^T) \quad \text{(A-8)}$$

where $\mu_p$ and $\mu_q$ are the respective means of $p$ and $q$, and $\Sigma_p$ and $\Sigma_q$ are the respective covariances of $p$ and $q$. The predicate `sig:jsdiv` used in this thesis identify the computation of a symmetrised Kullback-Leibler divergence bewteen two pairs of mean and covariance.

# Appendix B

# Describing Semantic Web datasets

We here describe our approach for characterising the content of a Semantic Web dataset. If we have muliple RDF datasets, in the form of multiple RDF docments available on the web and multiple SPARQL end-points, we want to know which one would be able to answer a particular query. For example, if we are looking for albums, corresponding playlists and tags, we want to know that the Jamendo end-point described in § 6.3.2 would be able to answer such a query.

We follow a simple approach based on reified RDF graphs examplifying the content of a particular dataset.

## B.1 Background

Previous work tried to solve this problem, in the same context. The SPARQL Service Advertisement and Discovery (SADDLE [Cla]) aimed at investigating a protocol to advertise the capabilities of a particular end-point. SADDLE used to be a part of the SPARQL recommendation draft. The RDF graph pattern and templates of the RIRO vocabulary [Ego06] can be used to describe a particular dataset, by describing the different RDF graph patterns it can satisfy. The middleware architecture described in [LWB08] is particularly interesting, as the dataset descriptions are generated automatically from the dataset itself. This description consists in a list of concepts along with the corresponding number of instances and a list of properties and their occurences. However, this automated description is based on aggregates (`COUNT(*)` statements), which are not currently part of SPARQL.

## B.2 Characterising datasets

Our approach to characterise the content of a particular RDF dataset is to publish a small example of the sort of information it holds. We consider an identifier `void:example`, linking a web resource identifying a dataset to an RDF document examplifying it. For example, the Jamendo dataset described in § 6.3.2 would be described as follows.

```
:ds1
  a void:Dataset;
  rdfs:label "Jamendo end-point on DBtune";
  dc:source <http://jamendo.com/>;
  foaf:maker <http://moustaki.org/foaf.rdf#moustaki>;
  void:sparql_end_point <http://dbtune.org:2105/sparql/>;
  void:example <http://moustaki.org/void/jamendo_example.n3>.
```

The `void:example` property links to an RDF document including the following statements.

```
<http://dbtune.org/jamendo/artist/5>
  a mo:MusicArtist;
  foaf:based_near <http://sws.geonames.org/2991627/>;
  owl:sameAs <http://dbtune.org/musicbrainz/resource/artist/
0781a3f3-645c-45d1-a84f-76b4e4decf6d>;
  foaf:img <http://img.jamendo.com/artists/b/both.jpg>;
  foaf:homepage <http://www.both-world.com>;
  foaf:name "Both";
  foaf:made <http://dbtune.org/jamendo/record/33> .

<http://dbtune.org/jamendo/record/33>
  a mo:Record;
  dc:date "2004-12-28 18:46:23";
  mo:available_as <http://www.jamendo.com/get/album/id/album/p2p/redirect/
33/?p2pnet=bittorrent&amp;are=mp32>;
  mo:image <http://img.jamendo.com/albums/33/covers/1.0.jpg>;
  mo:track <http://dbtune.org/jamendo/track/241>;
  tags:taggedWithTag <http://dbtune.org/jamendo/tag/alternatif>;
  owl:sameAs <http://dbtune.org/musicbrainz/resource/record/
fade0242-e1f0-457b-99de-d9fe0c8cbd57> .
```

This example RDF document holds information about the different types of resources and relationships we can find in this dataset.

## B.3 Finding a dataset

Now, when looking for a particular information, we use the Named Graphs [CBHS05a] feature of the SPARQL recommendation, allowing us to query the description of the dataset as well as the example linked from it. For example, the following SPARQL query returns the end-point of a dataset holding information about music records, associated tags, and places to download them.

```
SELECT ?sparql
FROM NAMED <http://moustaki.org/void/void.n3>
FROM NAMED <http://moustaki.org/void/jamendo_example.n3>
{
        GRAPH <http://moustaki.org/void/void.n3> {
                ?ds a void:Dataset;
                        void:sparql_end_point ?sparql;
                        void:example ?ex.
        }
        GRAPH ?ex {
                ?r a mo:Record;
                        mo:available_as ?l;
                        tags:taggedWithTag ?t.
        }
}
```

Such descriptions are small enough to be aggregated in a SPARQL end-point, able to redirect queries to the matching end-points using similar queries. The inputted graph pattern just needs to be part of the `GRAPH ?ex {...}` part of our query.

# Appendix C

# Automatic interlinking algorithm pseudo-code

The algorithm described in § 6.4.5 corresponds to the following pseudo-code. We want to automatically interlink two datasets $A$ and $B$. We assume the existence of a function $\mathsf{string\_similarity}(x, y)$ returning a similarity measure between two strings. Other similarity measures can be substituted for other types of resources, e.g. a content-based audio similarity. $G_x$ is the RDF graph available when accessing $x$. We define the following additional functions:

```
function $\mathsf{similarity}(x, y)$ :
    Extract a suitable label $l_x$ for $x$ in $G_x$
    Extract a suitable label $l_y$ for $y$ in $G_y$
    Return $\mathsf{string\_similarity}(l_x, l_y)$

function $\mathsf{lookup}(x)$ :
    Extract a suitable label $l_x$ for $x$ in $G_x$
    Perform a search for $l_x$ on $B$
    Return the set of resources retrieved from the search

function $\mathsf{measure}(M)$ :
    Foreach $(r_i, r_j) \in M$
        $\mathsf{sim}_{i,j} = \mathsf{similarity}(r_i, r_j)$
    Return $\sum_{i,j} \mathsf{sim}_{i,j}$

function $\mathsf{combinations}(O_1, O_2)$ :
    Return all possible combinations of
    elements of $O_1$ and elements of $O_2$
    e.g. $\mathsf{combinations}(\{1, 2\}, \{3, 4\}) = \{\{(1, 3), (2, 4)\}, \{(1, 4), (2, 3)\}\}$
```

Our starting point is a URI $a$ in $A$ and a decision threshold $\mathsf{threshold}$. Our automatic interlinking pseudo-code is then defined as:

```
Foreach $s_k \in \mathsf{lookup}(a)$
  $M_k = \{(a, s_k)\}$
  $\mathsf{measure}_k = \mathsf{measure}(M_k)$
  $\mathsf{sim}_k = \mathsf{measure}_k / |M_k|$
If $\mathsf{sim}_k > \mathsf{threshold}$ for exactly one $k$, **return** $M_k$
Else, Mappings is the list of all $M_k$, and **return** propagate(Mappings)

function propagate(Mappings) :
  Foreach $M_k \in$ Mappings:
    $\mathsf{measure}_k = \mathsf{measure}(M_k)$
    Foreach $p$ s.t. $(\exists(r, r') \in M_k,\ \exists(r, p, o) \in G_r,\ \exists(r', p, o') \in G_{r'}$ and $\forall \mathsf{Map} \in \mathsf{Mappings}, (o, o') \notin \mathsf{Map})$:
      Foreach $(r, r') \in M_k$:
        $O_{k,r,p}$ is the list of all $o$ such that $(r, p, o) \in G_r$
        $O'_{k,r,p}$ is the list of all $o$ such that $(r', p, o) \in G_{r'}$
      Foreach $\mathsf{Objmap}_{k,i} \in \bigcup_r \mathsf{combinations}(O_{k,r,p}, O'_{k,r,p})$:
        $\mathsf{sim}_{k,i} = (\mathsf{measure}_k + \mathsf{measure}(\mathsf{Objmap}_{k,i}))/(|M_k| + |\mathsf{Objmap}_{k,i}|)$
  If no $\mathsf{sim}_{k,i}$, **fail**
  If $\mathsf{sim}_{k,i} > \mathsf{threshold}$ for exactly one $\{k, i\}$ pair, **return** $\mathsf{append}(M_k, \mathsf{Objmap}_{k,i})$
  Else, NewMappings is the list of all $\mathsf{append}(M_k, \mathsf{Objmap}_{k,i})$, and **return** propagate(NewMappings) (if the maximum number of recursions is not reached, otherwise **fail**)
```

Now, we apply this pseudocode to the "Both" example in § 6.4.5 ($a = \underline{1} =$ `http://dbtune.org/jamendo/artist/5`), with a threshold of 0.8. This makes us go through the following steps:

```
$\mathsf{lookup}(r) = \{4, 7\}$
$M_1 = \{(1, 4)\}$, $\mathsf{sim}_1 = 1$
$M_2 = \{(1, 7)\}$, $\mathsf{sim}_2 = 1$
$\mathsf{Mappings} = \{\{(1, 4)\}, \{(1, 7)\}\}$

$\mathsf{propagate}(\{\{(1, 4)\}, \{(1, 7)\}\})$
  $k = 1$, $p = \mathsf{foaf{:}made}$
    $O_{1,1,\mathsf{foaf:made}} = \{2, 3\}$, $O'_{1,4,\mathsf{foaf:made}} = \{5, 6\}$
    $\mathsf{Objmap}_{1,1} = \{(2, 5), (3, 6)\}$, $\mathsf{Objmap}_{1,2} = \{(2, 6), (3, 5)\}$
    $\mathsf{sim}_{1,1} = 0.9$, $\mathsf{sim}_{1,2} = 0.4$
  $k = 2$, $p = \mathsf{foaf{:}made}$
    $O_{2,1,\mathsf{foaf:made}} = \{2, 3\}$, $O'_{2,7,\mathsf{foaf:made}} = \{8\}$
    $\mathsf{Objmap}_{2,1} = \{(2, 8)\}$, $\mathsf{Objmap}_{2,2} = \{(3, 8)\}$
    $\mathsf{sim}_{2,1} = 0.55$, $\mathsf{sim}_{2,2} = 0.55$
  Now, $\mathsf{sim}_{1,1}$ is the only $\mathsf{sim}_{k,i}$ above the threshold, we therefore choose $\{(1, 4), (2, 5), (3, 6)\}$ as our mapping, which corresponds to the RDF code in § 6.4.5.3.
```

# Appendix D

# First bar of Debussy's Syrinx

The following RDF describes the first bar of Debussy's Syrinx using the Symbolic Ontology extension of the Music Ontology framework described in § 3.3.4.3.

```
<>
  rdfs:comment "The first bar of Debussy's core of Syrinx";
  rdfs:label "syrinx" .

:compo
  a mo:Composition;
  event:product :syrinx_w;
  event:product :syrinx_s .
:syrinx
  a mo:MusicalWork;
  owl:sameAs <http://dbpedia.org/resource/Syrinx_\%28Debussy\%29> .
:syrinx_s
  a mo:Score;
  so:score_time :tl .


# Main timeline

:tl a so:ScoreTimeLine .

# Events

:e000
  a so:DottedEighthNote;
  so:pitch :b_flat;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e001;
    ] .
```

```
:e001
  a so:ThirtysecondNote;
  so:pitch :a;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e002 ] .
:e002
  a so:ThirtysecondNote;
  so:pitch :b;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e003 ] .
:e003
  a so:DottedEighthNote;
  so:pitch :a_flat;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e004 ] .
:e004
  a so:ThirtySecondNote;
  so:pitch :g;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e004 ] .
:e005
  a so:ThirtySecondNote;
  so:pitch :a;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e006 ] .
:e006
  a so:SixteenthNote;
  so:pitch :g_flat;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e007;
  ] .
:e007
  a so:SixteenthNote;
  so:pitch :f;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e008 ] .
:e008
```

```
  a so:SixteenthNote;
  so:pitch :e;
  event:time [
    tl:onTimeLine :tl;
    tl:meets :e009 ] .
:e009
  a so:SixteenthNote;
  so:pitch :d_flat;
  event:time [
    tl:onTimeLine :tl ] .

# Articulations

:s001
  a so:Slur;
  event:sub_event :e000;
  event:sub_event :e001;
  event:sub_event :e002 .
:s002
  a so:Slur;
  event:sub_event :e003;
  event:sub_event :e004;
  event:sub_event :e005 .
:s003
  a so:Slur;
  event:sub_event :e006;
  event:sub_event :e007;
  event:sub_event :e008;
  event:sub_event :e009 .


# Pitches

:b_flat
  a so:Pitch;
  so:note so:b;
  so:accidental so:flat;
  so:octave "1" .
:a
  a so:Pitch;
  so:note so:a;
  so:accidental so:natural;
  so:octave "1" .
:a_flat
```

```
  a so:Pitch;
  so:note so:a;
  so:accidental so:flat;
  so:octave "1" .
:g
  a so:Pitch;
  so:note so:g;
  so:accidental so:natural;
  so:octave "1" .
:g_flat
  a so:Pitch;
  so:note so:g;
  so:accidental so:flat;
  so:octave "1" .
:f
  a so:Pitch;
  so:note so:f;
  so:accidental so:natural;
  so:octave "1" .
:e
  a so:Pitch;
  so:note so:e;
  so:accidental so:natural;
  so:octave "1" .
:d_flat
  a so:Pitch;
  so:note so:d;
  so:accidental so:flat;
  so:octave "1" .
```

# Appendix E

# Namespaces

The following namespaces are used throughout this thesis.

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix mo: <http://purl.org/ontology/mo/> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix tl: <http://purl.org/NET/c4dm/timeline.owl#> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix event: <http://purl.org/NET/c4dm/event.owl#> .
@prefix af: <http://purl.org/ontology/af/> .
@prefix co: <http://purl.org/ontology/chord/> .
@prefix so: <http://purl.org/ontology/symbolic-music/> .
@prefix pop: <http://purl.org/ontology/pop-structure/> .
@prefix mit: <http://purl.org/ontology/mo/mit/> .
@prefix time: <http://www.w3.org/2006/time#> .
@prefix skos: <http://www.w3.org/2004/02/skos/core#> .
@prefix po: <http://purl.org/ontology/po/> .
@prefix fb: <http://www.daml.org/2003/09/factbook/factbook-ont#> .
@prefix math: <http://www.w3.org/2000/10/swap/math#> .
@prefix list: <http://www.w3.org/2000/10/swap/list#> .
@prefix log: <http://www.w3.org/2000/10/swap/log#> .
@prefix ctr: <http://purl.org/ontology/ctr/> .
@prefix img: <http://purl.org/ontology/img/> .
@prefix sim: <http://purl.org/ontology/sim/> .
@prefix sig: <http://purl.org/ontology/sig/> .
@prefix rel: <http://purl.org/vocab/relationship/> .
@prefix wgs: <http://www.w3.org/2003/01/geo/wgs84_pos#> .
@prefix geo: <http://www.geonames.org/ontology#> .
@prefix tags: <http://www.holygoat.co.uk/owl/redwood/0.1/tags/> .
```

```
@prefix yago: <http://dbpedia.org/class/yago/> .
@prefix p: <http://dbpedia.org/property/> .
@prefix pc: <http://purl.org/ontology/playcount/> .
@prefix awol: <http://bblfish.net/work/atom-owl/2006-06-06/#> .
@prefix qvp: <http://purl.org/ontology/qvp/> .
@prefix gnat: <http://motools.sourceforge.net/doap.rdf#> .
@prefix : <http://example.org/> .
@prefix i1: <http://example.org/interpreter1/> .
@prefix i2: <http://example.org/interpreter2/> .
```