Piotr Adam Praczyk

Management of Scientific Images: an approach to the extraction, annotation and retrieval of figures in the field of High Energy Physics

ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

Tesis Doctoral

MANAGEMENT OF SCIENTIFIC IMAGES: AN APPROACH TO THE EXTRACTION, ANNOTATION AND RETRIEVAL OF FIGURES IN THE FIELD OF HIGH ENERGY PHYSICS

Autor

Piotr Adam Praczyk

Director/es

Mele, Salvatore
Nogueras Iso, Javier

UNIVERSIDAD DE ZARAGOZA
Informática e Ingeniería de Sistemas

2013

Management of Scientific Images: an approach to the extraction, annotation and retrieval of figures in the field of High Energy Physics

Piotr Adam Praczyk

PhD DISSERTATION

RESEARCH ADVISORS

Dr Javier Nogueras-Iso

Dr Salvatore Mele

September 2013

Computer Science and Systems Engineering Department

Universidad de Zaragoza

CERN

Acknowledgements

Writing a thesis is impossible without the participation and support of other people. I would like to express my gratitude towards everyone who has made this project possible.

First of all, I would like to thank my research advisors, dr Javier Nogueras-Iso and dr Salvatore Mele for their insight, continuous support and encouragement. I have experienced much more help and patience that I could wish for.

From an institutional perspective, I would like to thank the support provided by the Scientific Information Service at CERN (for funding my student grant), the Advanced Information Systems Group (IAAA) of the Computer Science and Systems Engineering Department at the University of Zaragoza, and the Spanish Government through the national research project TIN2012-37826-C02-01.

I would like to thank all my colleagues from the INSPIRE collaboration and from the Invenio project for sharing their knowledge and for their suggestions which influenced this work. Special thanks to dr Mike Whalley from the University of Durham for all the input and help when working on the integration of datasets from HepData. I am particularly indebted to Peter Hofmann and Daniel Stanculescu who helped with the implementation. Peter worked on improving the manual figures extractor and integrated it with Invenio. Daniel implemented the module for merging the metadata coming from different sources. I am also very grateful to the members of the IAAA group of the University of Zaragoza for a very warm reception, help and suggestions every time I was visiting the university.

Working at CERN was a great adventure which influenced me both personally and professionally. I would like to thank all my friends who I have met during this time and who became an integral part of my life. This time would not be as unique without your presence and all the things we were doing together. Thank you Roman, Hugo, Ruben, Michał, Nicola, Victor, Jan, Kasia, Thorsten (for being the most reliable and pushing climbing partner I have ever had), Tomek, Simon, Netko, Laura, Samuele, Ina, Joe, Annette, Thomas, Valkyrie, Nicoletta, Suenje, Patricia and many more.

Last, but not least, I would like to thank my family, especially my parents and brother for their constant support and encouragement during the entire time of writing.

There are many more people who have contributed to the writing of this dissertation and are not listed here. I would like to thank you all!

Piotr Praczyk
Zaragoza, September 2013

Abstract

The information environment of the first decade of the XXIst century is unprecedented. The physical barriers limiting access to the knowledge are disappearing as traditional methods of accessing information are being replaced or enhanced by computer systems. Digital systems are able to manage much larger sets of documents, confronting information users with the deluge of documents related to their topic of interest. This new situation created an incentive for the rapid development of Data Mining techniques and to the creation of more efficient search engines capable of limiting the search results to a small subset of the most relevant ones. However, most of the up to date search engines operate using the text descriptions of the documents. Those descriptions can either be extracted from the content of the document or be obtained from the external sources. The retrieval based on the non-textual content of documents is a subject of ongoing research. In particular, the retrieval of images and unlocking the information carried by them attracts a lot of attention of the scientific community.

Digital libraries hold a special position amongst the systems allowing the access to knowledge. They serve the role of repositories of documents which share some common characteristics (e.g. belonging to the same area of knowledge or produced in the same institution) and as such, contain documents selected as interesting for a particular group of users. In addition, they provide retrieval facilities on top of the managed collections.

Typically, scholarly publications are the smallest units of information managed in scientific digital libraries. However, there are different types of artifacts produced and used in the scientific process, among others: figures and datasets. Figures play a particularly important role in the process of scholarly publishing. Representing data in a graphical manner allows showing patterns in large datasets and to make complicated ideas easier to understand. The existing digital library systems enable the access to figures only as part of the files used for the serialisation of the entire publication.

The objective of this thesis is to propose a set of methods and techniques in order to transform figures into first-class products within the scientific publication process, allowing researchers to get the maximum benefit from the search and review of bibliography. The proposed methods and techniques are oriented towards the acquisition, semantic annotation and search of figures contained in scholarly publications. Leveraging the completeness of the field and the existing community, we illustrated the described theory with examples from High-Energy Physics (HEP). At every place requiring more focused considerations, we concentrated on the type of figures that appear more frequently in the corpus of HEP publications: the plots. The described prototypes capable of processing figures have been partially integrated with the

Invenio[1] digital library software and INSPIRE - one of the largest digital libraries in the world on High-Energy Physics and created by the collaboration of the main laboratories and research centres in this domain (CERN, SLAC, DESY and Fermilab).

[1]`http://invenio-software.org/`

Resumen

El entorno de la información en la primera década del siglo XXI no tiene precedentes. Las barreras físicas que han limitado el acceso al conocimiento están desapareciendo a medida que los métodos tradicionales de acceso a información se reemplazan o se mejoran gracias al uso de sistemas basados en computador. Los sistemas digitales son capaces de gestionar colecciones mucho más grandes de documentos, confrontando a los usuarios de información con la avalancha de documentos asociados a su tópico de interés. Esta nueva situación ha creado un incentivo para el desarrollo de técnicas de minería de datos y la creación de motores de búsqueda más eficientes y capaces de limitar los resultados de búsqueda a un subconjunto reducido de los más relevantes. Sin embargo, la mayoría de los motores de búsqueda en la actualidad trabajan con descripciones textuales. Estas descripciones se pueden extraer o bien del contenido o a través de fuentes externas. La recuperación basada en el contenido no textual de documentos es un tema de investigación continua. En particular, la recuperación de imágenes y el desentrañar la información contenida en ellas están suscitando un gran interés en la comunidad científica.

Las bibliotecas digitales se sitúan en una posición especial dentro de los sistemas que facilitan el acceso al conocimiento. Actúan como repositorios de documentos que comparten algunas características comunes (por ejemplo, pertenecer a la misma área de conocimiento o ser publicados por la misma institución) y como tales contienen documentos considerados de interés para un grupo particular de usuarios. Además, facilitan funcionalidades de recuperación sobre las colecciones gestionadas.

Normalmente, las publicaciones científicas son las unidades más pequeñas gestionadas por las bibliotecas digitales científicas. Sin embargo, en el proceso de creación científica hay diferentes tipos de artefactos, entre otros: figuras y conjuntos de datos. Las figuras juegan un papel particularmente importante en el proceso de publicación científica. Representan los datos en una forma gráfica que nos permite mostrar patrones sobre grandes conjuntos de datos y transmitir ideas complejas de un modo fácilmente entendible. Los sistemas existentes para bibliotecas digitales facilitan el acceso a figuras, pero solo como parte de los ficheros sobre los que se serializa la publicación entera.

El objetivo de esta tesis es proponer un conjunto de métodos y técnicas que permitan transformar las figuras en productos de primera clase dentro del proceso de publicación científica, permitiendo que los investigadores puedan obtener el máximo beneficio a la hora de realizar búsquedas y revisiones de bibliografía existente. Los métodos y técnicas propuestos están orientados a facilitar la adquisición, anotación semántica y búsqueda de figuras contenidas en publicaciones científicas. Para demostrar la completitud de la investigación se han ilustrado las

teorías propuestas mediante ejemplos en el campo de la Física de Partículas (también conocido como Física de Altas Energías). Para aquellos casos en los que se han necesitado propuestas más detalladas, esta tesis se ha focalizado en las figuras que aparecen con más frecuencia en las publicaciones de Física de Partículas: los gráficos científicos denominados en inglés con el término *plots*. Los prototipos que se han desarrollado para esta tesis se han integrado parcialmente dentro del software Invenio[2] para bibliotecas digitales, así como dentro de INSPIRE, una de las mayores bibliotecas digitales en Física de Partículas mantenida gracias a la colaboración de grandes laboratorios y centros de investigación como son el CERN, SLAC, DESY y Fermilab.

[2]`http://invenio-software.org/`

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Digital library systems are the natural candidates for storing and managing various types of artifacts related to the scholarly publishing. They originated as computer systems intended to replace the paper-card based catalogues of traditional libraries. Over the years of their development they have been gradually extended with the capabilities of browsing and searching. With the advent of the methods of digital publishing, they also started serving as repositories granting access to the digital version of the described publications.

Nowadays, in certain areas of knowledge, the content managed by the digital libraries has completely replaced physical copies of the documents. As such, digital libraries started serving a new important role of being preservation platforms for entire branches of the human knowledge. In particular, this created requirements of not only storing the metadata and documents, but also of creating workflows assuring the persistence and the availability of the data - features seldom visible to the users, but crucial for the preservation of the human knowledge. Digital libraries also started playing an important role in the process of learning and discovering the most recent scientific results by providing instant access to a large number of documents. The rapid development of digital publishing and preservation techniques also removed some of the barriers of creating scientific content. It also created the incentive to question and rethink many aspects of the traditional publishing - the questions of verification of the provided content and, as the geographical barriers were no more existent, of the access to the resources.

Additionally, in recent years digital libraries have shifted their focus towards a more complete description of publishing artifacts. Formerly, only publications were considered as first-class citizens of the publishing ecosystem. Now, the increasing number of efforts is made to give access to figures, tables and datasets being part of the publications. In particular, Figures play an important role in the process of scholarly publishing. Representing data in a graphical manner allows showing patterns in large datasets and to make complicated ideas easier to understand. Having all those artefacts gathered in a single place and annotated in a uniform manner introduced completely new possibilities of linking between those different types of research results. The structure of relations between those objects can reflect both the scientific process behind their creation and the relation at a more conceptual level.

Furthermore, the information environment of the first decade of the XXIst century is unprecedented: a deluge of born-digital content, ubiquitous and searchable, where the Web2.0 paradigm blurs the distinction between information producer and consumer. Current digital libraries interoperate with other infrastructures and establish connections with social web or any other resource on the Web.

This movement is also noticeable in the context of digital libraries for High Energy Physics (HEP), the application domain where the theoretical contributions of this PhD are applied. HEP, also called Particle Physics concentrates on describing the smallest particles being the building blocks of the surrounding reality. The human understanding of what are the smallest building blocks of the matter has changed over centuries starting with the ideas of the ancient Greek philosophers introducing the idea of elements. At present, the smallest known particles of subatomic sizes are described by the Standard Model. There exist different theories which still need to be tested. Large physics experiments conducted in the laboratories as European Organisation for Nuclear Research (CERN) or Stanford Linear Accelerator Center (SLAC) test the viability of these theories and measure the exact properties of the already discovered particles. Particle Physics is a large branch of modern science which has developed its own methodologies and practices. Due to the importance of research on HEP and the investments done in this area, we must remark as well the great efforts done in research and innovation for scientific digital libraries in this domain.

As a verification of the relevance granted for digital libraries in this domain, we can state that the history of these libraries interlaces with the history of the Web. In 1990, Tim Berners Lee[1] (working at the time at CERN - European Organisation for Nuclear Research) wrote the first proposal of World Wide Web (WWW) and implemented the first web server. In 1991, Stanford Public Information Retrieval System (SPIRES) - the digital library of preprints from High-Energy Physics was extended with the web interface and became the first website in America. Both the SPIRES back-end system (written in 70s) and the web interface had over the years became difficult to maintain and to extend. In recent years, SPIRES has been replaced by INSPIRE, which is based on a much more modern software platform (Invenio) and contains the complete content which was originally managed by SPIRES. This new system aims to provide tens of thousands of scientists with a million records enriched by full-text searches, specialist curation, bibliometric analysis and an increasing attention to the potential of the web2.0 paradigm to foster scientific communication. Unlike SPIRES, which was maintained and updated at the SLAC laboratory, INSPIRE is run by a larger collaboration including large

[1]`http://www.w3.org/History/1989/proposal-msw.html`

physics laboratories like CERN, SLAC, Deutsches Elektronen-Synchrotron (DESY) or Fermi Lab, INSPIRE is the main platform of application of the work described in this thesis.

Currently, INSPIRE contains over 1,000,000 records describing publications deposed on arXiv[2], published in journals, manually submitted by the authors and coming from other sources. The surveys [27] have shown that INSPIRE is the main information resource for the majority of the particle physics community. The recent developments of INSPIRE concentrate on providing more intelligent tools allowing the manipulation of the data. Among others, it is worth to mention the automatic author disambiguation (supported by a distributed data curation using crowdsourcing tools). INSPIRE also begins to be used as a data preservation platform, which allows the storage of different types of research-oriented objects.

The investigation on the content of images is the next frontier in INSPIRE. The graphical content included in scholarly publications allows a much more efficient access to the most important results presented in a publication[26, 29]. The human brain perceives the graphical content much faster than reading an equivalent block of text. Presenting figures together with the publication summary, when displaying search results, would allow a more accurate assessment of the article content and in turn lead to a better usage of researchers time. Enabling users to search for figures describing quantities or phenomena similar to a given one could become a very powerful tool for finding publications describing similar results. Combined with additional metadata, it could provide knowledge about evolution of certain measurement or idea over time. These and many more applications created an incentive to research possible ways integration of figures in INSPIRE.

The problem is that the processing of image content in scholarly publication is not an immediate task given the volume and heterogeneity of electronic materials and formats that are stored nowadays in digital libraries. This problem is not only specific to digital libraries or HEP digital libraries. Quite the opposite, it is a common problem that is general to the processing of all the data on the Web. Yet, not all the data created, encoded or stored using computers can be automatically processed and reasoned about using only the machines. This scenario constitutes an opportunity and a challenge for the management of scientific knowledge. The next frontier in the IT industry is clearly at the crossroad of the "Semantic Web" and the "Web of Data".

A possible line to facilitate the processing and integration of images for information retrieval of scholarly publications is the application of semantic technologies. Understanding the meaning of the content and unlocking the data already present in scholarly communication can produce further and relevant information. With the advent of the techniques of the Semantic Web, digital libraries were equipped with techniques facilitating the description of their resources in a machine-readable form and also the establishment of the aforementioned links [19, 33].

Recent years have witnessed much research in the area of diminishing the semantic gap

[2]http://arxiv.org

related to the multimedia content - developing methods allowing computers to automatically generate the semantic annotation of multimedia. In most cases, methods of automatic annotation of photographs have been investigated. In the case of the automatic content extraction, two general methodologies are used. The first involves the discovery of the connections between data elements based on machine learning without much a priori knowledge about the content. Another manner of mining the multimedia data uses the knowledge of the expected structure of the image and of the field to which images are related.

However, despite multiple approaches to the automatic image content analysis, less research has been conducted in the area of processing the scholarly figures. To date, there is not a large scale application in digital libraries of semantic analysis and perusal of semantic information describing scientific images.

1.2 Objectives

The major goal of the Ph.D. research is the investigation of methods suitable for the extraction, description, and retrieval of information obtained from scientific images in digital libraries of scholarly publications. The specific objectives of the research include:

- To research and develop methods of the extraction of figures from the scholarly publications encoded in popular formats.
- To research and develop methods based on domain-specific knowledge to extract data and semantics from scientific figures.
- To investigate the possible ways to integrate the semantic engines within the strategies of search engines.
- To deal with problems related to the storage and processing of high volumes of data in a digital library.

The above objectives cover the main challenges related to managing figures as objects in a digital library. Their achievement paves the road for additional more complex applications and shows the direction of future research.

1.3 Scope

The scope of this thesis is to study the problem of processing figures in the environment of the INSPIRE digital library. We investigate the methods of retrieving figures encoded in scholarly publications of HEP, which are managed by INSPIRE. In particular, we describe the prototype of the PDF document analysis framework which has been implemented and the methods of integrating this prototype as a part of the INSPIRE data acquisition workflow. Our interest focuses on the description of the semantics of figures both in relation to other publishing artefacts (for example the publication in which a figure is included) and of the internal meaning of the image itself. Special importance is assigned to the most frequent types of scholarly figures - the plots. We also research the methods of automatically generating such annotation of automatically extracted figures.

The scope of the present research also includes the investigation of how to retrieve an image corresponding to a given concept so to enable its further reuse in the scholarly process. We describe the experiments with a sample set of annotated figures and elaborate a method of integrating such a search in the INSPIRE digital library. We also investigate methods of using the semantic annotation to compare the figures among themselves.

1.4 The Structure of the Thesis

The remainder of this thesis is divided into chapters corresponding to steps necessary to provide semantic annotation of figures.

- Chapter 2 presents a formal vocabulary (a HEP Figures Ontology), which allows the definition of different types of figures as well as other artifacts in the scholarly publication process. This vocabulary also allows the annotation of figures.
- Chapter 3 describes a novel method of analysing the content of the publications encoded as PDF files and extracting the contained figures.
- Chapter 4 deals with automatic extraction of the content of plots, which are the most ubiquitous type of figures appearing in HEP publications.
- Chapter 5 describes the method of using the semantic annotation for searching.
- Chapter 6 describes the integration of the previously described methods and prototypes inside INSPIRE. The description comprises a more technical considerations of the Invenio Digital Library system, the modifications to the underlying data model which allowed better description of figures, INSPIRE-specific methods of acquiring figures, Integration with the HepData system and integration of the semantic annotation of figures with the internal Invenio search engine.
- Chapter 7 summarises the contribution of the thesis and describes the possible directions of future development.

Chapter 2

HEP Figures Ontology

2.1 Introduction

Being able to process the internal semantics of figures requires first having a method of describing their structure and meaning. This chapter discusses one of the main components of a system processing figures - describing the internal semantics using a formal vocabulary. We provide the ontology of different artefacts related to publications of High-Energy Physics (HEP).

When designing the ontology, our main objective was to describe the information about the structure and the content of figures while abstracting from the less relevant details of their particular representation. The main purpose of this type of metadata is to allow the semantic discovery of figures.

The additional objective consisted of describing the representation details of the abstract entities related to the principal objective. Usages of this type of metadata can include automatic processing of figure images (combining several of them in a single image, highlighting parts of the content or generating figures containing only part of the metadata). This type of metadata can appear as a side effect of generating more abstract description from the unstructured figure file.

The role of a graphical illustration transcends pure representation of the data. Graphs are designed to be read and interpreted by humans and as such, they contain elements trying to guide the interpretation of the data. Certain properties of the data are usually emphasised using various visual techniques [19]. For instance, the choice of scales used on axes (linear, logarithmic etc.) can be used to make some patterns inside a dataset more visible. In the context of HEP, also the description of supplementary graphical properties can play an important role during the interpretation of the data. The usage of colours can be meaningful, indicating the type of a plot (e.g. a blue band plot usually describes the same type of phenomena and the colour is used as an identification of a figure type). All those elements had to be considered during the design and development of the ontology.

Semantic technologies have been successfully applied to increase the visibility and discovery in other related domains such as general analysis of statistical data [19] or accessibility of bibliographic databases [33]. Following this strategy, the objective of this chapter is to propose a semantic approach for the annotation of the different artefacts involved in the scholarly publications of HEP research. We propose an application ontology, called HEP Figures Ontology (HFO), which is based on existing domain ontologies. Although Hep Figures Ontology (HFO) allows describing arbitrary types of HEP figures, special attention has been placed on describing the internal structure of plots - the most frequent type of HEP figures.

The semantic annotation has in recent years been an object of intensive research and standardisation. At the moment of writing, the two most popular semantic annotation standards are Resource Description Framework (RDF) and Web Ontology Language (OWL) [3]. RDF is a metadata model which defines all the annotation as a series of statements, which describe properties of different objects and relations between them. Every statement takes the form of a triple (*Subject*, *Predicate*, *Object*), for example (*picture*, *contains*, *tree*). RDF does not define a particular method of encoding the statements, providing only the abstract definition of the data model. There exist multiple methods of encoding RDF, out of which Extensible Markup Language (XML) is the most popular. OWL is a language based on RDF. It defines a number of predicates together with their meaning. The defined predicates enable for example describing that two entities are the same. OWL also allows describing different reasoning rules which together with predefined predicates facilitate the automatic generation of new true statements about the dataset. There exist different types of OWL (e.g. OWL Lite, OWL DL), which correspond to different models of the Description Logic (DL) [4]. The remainder of this chapter is organised as follows: Section 2.2 presents the existing attempts aiming at the description of figures and datasets in both HEP and in other areas. Section 2.3 describes the structure of HFO. Finally, this chapter ends with some conclusions and outlook on future work.

2.2 Related Work

There are formal vocabularies allowing the description of concepts in High-Energy Physics (HEP) such as the HEP taxonomy [39], which is an Simple Knowledge Organization System (SKOS)-based vocabulary for the classification of INSPIRE publications, or the compilation of properties of elementary HEP particles published by the Particle Data Group (PDG)[1]. However, according to our knowledge, there are no ontologies for the description of figures in this specific domain. Therefore, this section reviews some works that intersect partially with our interest in defining a model for the description of figures in this domain.

HepData[2] [44] is an example of a project gathering and allowing access to the data behind

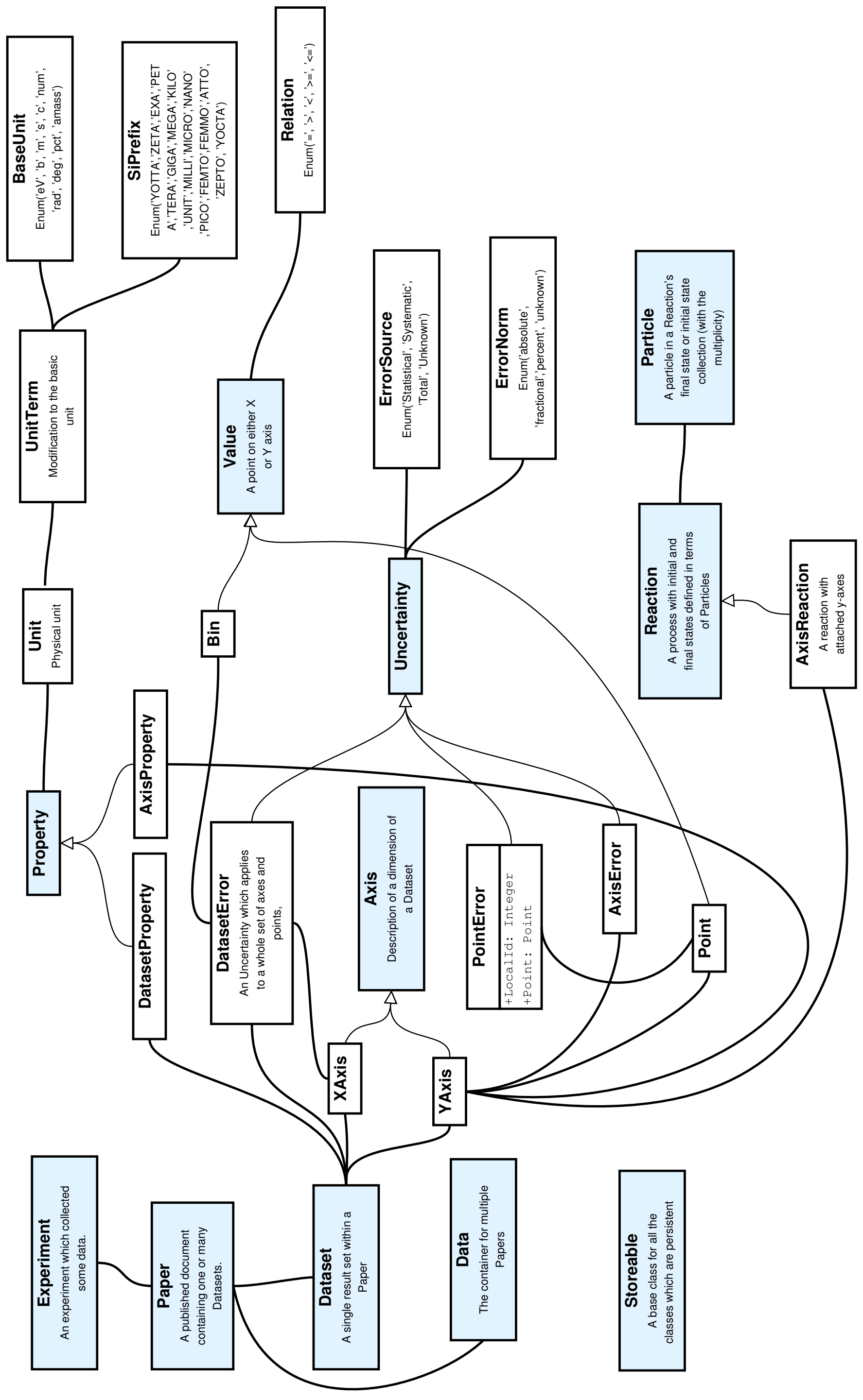
[1]`http://pdg.lbl.gov/2012/html/what_is_pdg.html`
[2]`http://hepdata.cedar.ac.uk/`

publications, in particular behind tables and figures. The data stored on HepData servers comes either from authors of the described publications or from a manual analysis of the figure content. HepData stores information about over 7000 manually selected scholarly publications. Most of the data corpus consists of results of Monte Carlo (MC) simulations encoded in the form of histograms. This is reflected in the structure of the underlying data model (depicted in Figure 2.1), which is too specific for our purpose of describing general figures, not only those focused on MC simulations. An emphasis of HepData designers has been placed on encoding different types of errors which are related to a dataset (systematic, statistical). The internal data model of HepData contains a part responsible for encoding units, which are linked with the PDG identifiers, providing certain level of semantic interoperability with PDG and other databases.

Making use of semantic technologies, Cyber-Share [24] aims at providing a platform facilitating the documentation of the entire scientific process. For each research project supported by CyberShare, unformatted sources are processed to generate RDF annotations. Additionally, OWL ontologies are generated automatically using the vocabulary and properties found in RDF annotations. Although the proposed workflow and some technologies could be reused in the context of HEP, the ontologies generated by CyberShare are specific for each research project annotations. However, our aim is to define an application ontology for the common description of figures.

Dumontier et al. [19] created an ontology for describing statistical graphs. The objectives of their work varied from ours. Their principal goal was to build a system allowing access to scholarly graphs by people with sight disabilities. The emphasis of their work was placed on the description of the structure of statistical graphs, considering the visual design decisions taken by the creator. Their approach enables also the description of the data encoded in a graph. However, their ontology ignores the relation between the content and its visual representation. The second of their objectives was to perform searching for particular data values encoded in the graphs. The database used for experiments was constructed from the original spreadsheets used to create graphs, allowing the datasets to be described in terms of absolute numerical values. However, in general in a digital library we are in a different situation because we cannot obtain access to the original sources of all the data and data series associated to figures cannot be described in terms of numerical values. For this reason, we can only obtain data by the means of manual or automatic back-engineering. As such, our data has higher level of error which we should consider when constructing the search engine.

An interesting work by Samadian et al. [48] provides insight into the process of semantically annotating the results of medical examinations. The area of the work is completely different from ours. However, the techniques used by them can be generalised and reused in the context of HEP. Authors of this publication describe a complex information system storing a number of simple medical measurements (gathered over time in different places and under different



BaseUnit
Enum('eV', 'b', 'm', 's', 'c', 'num', 'rad', 'deg', 'pct', 'amass')
SiPrefix
Enum('YOTTA','ZETA','EXA','PETA','TERA','GIGA','MEGA','KILO','UNIT','MILLI','MICRO','NANO','PICO','FEMTO','FEMMO','ATTO','ZEPTO', 'YOCTA')
Relation
Enum('=', '>', '<', '>=', '<=')
UnitTerm
Modification to the basic unit
Value
A point on either X or Y axis
ErrorSource
Enum('Statistical', 'Systematic', 'Total', 'Unknown')
ErrorNorm
Enum('absolute', 'fractional','percent', 'unknown')
Particle
A particle in a Reaction's final state or initial state collection (with the multiplicity)
Unit
Physical unit
Bin
Uncertainty
Reaction
A process with initial and final states defined in terms of Particles
AxisReaction
A reaction with attached y-axes
AxisProperty
Property
DatasetProperty
DatasetError
An Uncertainty which applies to a whole set of axes and points,
Axis
Description of a dimension of a Dataset
PointError
+LocalId: Integer
+Point: Point
AxisError
Point
XAxis
YAxis
Experiment
An experiment which collected some data.
Paper
A published document containing one or many Datasets.
Dataset
A single result set within a Paper
Data
The container for multiple Papers
Storeable
A base class for all the classes which are persistent

Figure 2.1: The graph of entities used in the internal HepData data model. Blue rectangles describe subclasses of the Storable class

conditions). Those measurements are described using semantic techniques. The gathered observations are used to derive more complex knowledge. The system uses automatic reasoning because the knowledge-derivation rules vary in time, reflecting the progress of the medical knowledge. Some facts are difficult to derive using the standard reasoning techniques. The system implements semantic web-services which derive knowledge by the means of algorithmic techniques. Also in the case of scientific graphs, there are a high number of basic properties, which in turn can be used to derive more complex facts. As such, the work presented in [48] is very relevant when designing an information system for HEP. Part of the described vocabulary can also be used when annotating figures. The Semantic Science Integrated Ontology (SIO) [3] contains concepts of different types of figures. This description is however too simplistic to be used in our scenario. The work discusses also the semantic annotation of different types of units, which is much more complete than the one needed for HEP data associated with figures.

Encoding data used in Monte Carlo simulations in a reusable manner is essential for the HEP community [7]. There exist several standards like Les Houches Event File (LHEF) [2] or its later extension, HepML [7][4] providing a method of describing simulations. Those standards are used to exchange data between software components performing different stages of the simulation and as such, their use case is similar to describing figures by others. Experiences of those projects are valuable when designing a data models for describing figures. Besides describing MC simulations, there also exist standards for encoding different types of scientific results. For instance, FeynML [5] provides a prototype method of encoding Feynman diagrams using the XML syntax.

2.3 HEP Figures Ontology (HFO)

The HEP Figures Ontology (HFO) consists of several modules. The central module provides a framework for the description of the meaning and the structure of figures (and their associated scholarly objects) in the most abstract possible way. Additionally, in the case of describing specific figures such as plots, there are two other modules for the descriptions of coordinate systems, and data lines and areas. Last, there is another module of devoted to the description of material properties of figures. By the material description, we understand all the metadata that connects logical elements of a figure with their positions on the canvas of the encoding graphical file. These modules are explained in the following subsections. The ontology is accessible on-line in OWL format [6].

[3]`http://semanticscience.org`
[4]`http://hepml.hepforge.org`
[5]`http://feynml.hepforge.org`
[6]`https://github.com/ppiotr/InvenioSemantics/blob/master/files/inveniomodel.owl`

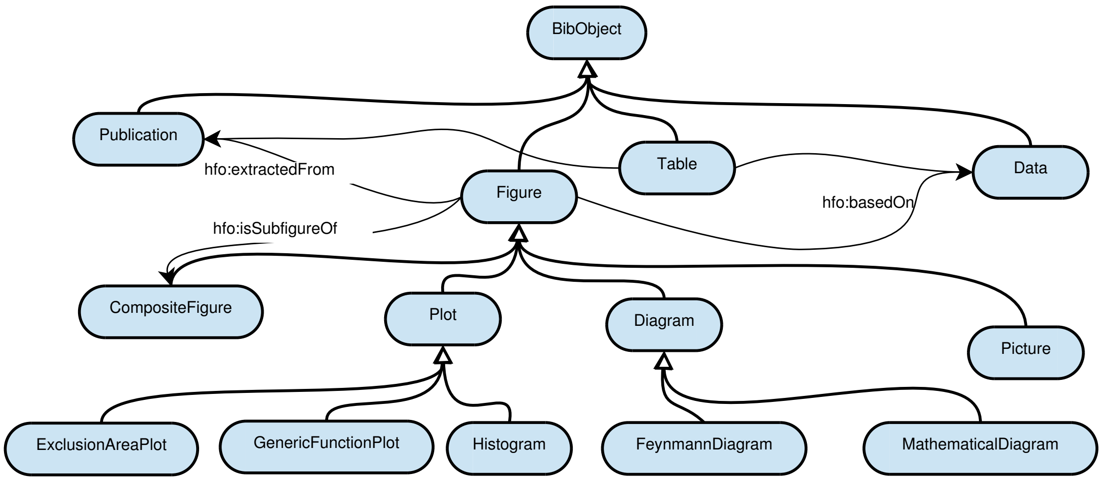


Figure 2.2: The central part of HFO

2.3.1 Central Concepts in HFO

Figure 2.2 depicts the taxonomy of some of the most important artefacts of scholarly publishing, which lies at the heart of HFO. Figures and tables form the centre of interest of HFO and, as such, they are classified in a much higher detail than publications or data. The taxonomy of different types of figures has been constructed by reviewing a corpus of available HEP publications and identifying the most common types. The most frequently appearing types of figures include plots, diagrams and general pictures (photographs and logos). Appendix A shows examples of different types of physics figures.

Two principal categories of figures (plots and diagrams) correspond to two general purposes which they are expected to serve. Plots tend to show the quantitative information by presenting numerical datasets in a way which facilitates their interpretation. Diagrams tend to provide a qualitative description, showing ideas and logical connections between them. Both those categories appear in HFO as specialisations of the Figure class.

Plots and tables share an important characteristic of presenting numerical data in a structured manner. As such, they can be seen as graphical representations of a certain dataset. With the advent of data preservation initiatives, digital libraries started treating not only publications, but also datasets as objects of the first category. When describing figures and tables, we can create links to appropriate publications from which those objects have been extracted. Similarly, if possible, we establish a connection between a plot and a dataset. It is worth noting that we can have many distinct plots representing the same dataset. A more detailed information about the treatment of datasets in the context of INSPIRE can be found in Chapter 6, in the section describing the integration with HepData.

The most common types of a plot present in HEP publications include histograms, which show in a graphical form how many times a measured value has fallen in a certain parametric range. Besides histograms, we consider exclusion plots depicting areas in a two-dimensional space. Meaning of those areas is usually understandable by the analysis of the context. HFO describes also general function plots showing a dependency graph of one parameter on another. Visually, general function plots can be very similar to histograms, however they carry a completely distinct semantics.

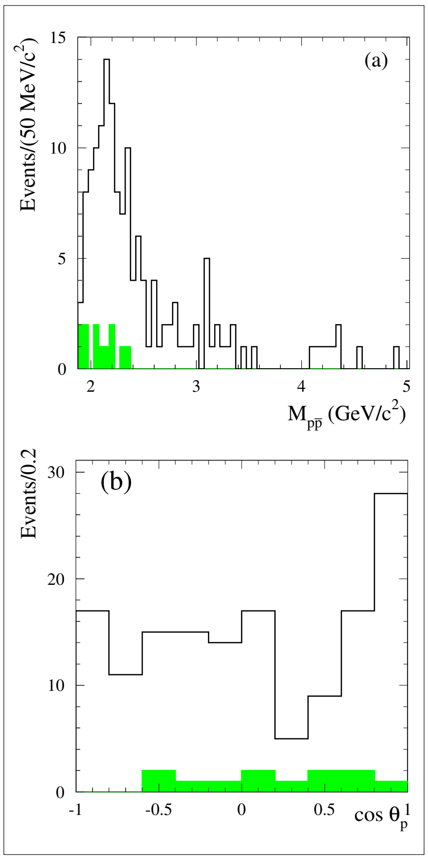
We have identified the most frequently used types of diagrams and included them in our taxonomy of publication artefacts. Some of the most important diagrams appearing in HEP publications are Feynman Diagrams. They encode in a schematic way a series of timely interactions between particles. Those interactions lead to annihilation of existing particles and to the creation of new ones. Particles are represented by lines (having different graphical characteristics) and their interactions by connection points between them. Also the generic mathematical diagrams appear in some of the physics publications. Those diagrams represent relations between abstract concepts and are typically characterised by having a number of nodes (represented by textual labels) connected with lines or arrows.

Sometimes, especially in the experimental setting, publications contain photographs or schemas of the equipment used to perform the measurement. Also different types of drawings or generic photographs can appear. Describing the internal meaning of those graphics lies outside of the scope of our work. However, there exist efforts to describe meaning of general images. In HFO, the description of custom images is limited to annotating them with the Picture class.

Creating a comprehensible taxonomy of all possible types of graphics was not our goal. Authors of scholarly publications typically follow a set of common practices of creating figures. People tend to create representation similar to those which they have already seen. However, there are no strict rules limiting what can and what cannot be considered a figure. The hierarchical structure of concepts in HFO allows providing the possibly most accurate description by instantiating a (possibly non-leaf) concept. For example a generic plot, or a diagram which does not fit into any of the predefined categories, can be annotated with the type of Plot or Diagram.

Sometimes many plots or other graphics form a single figure. HFO describes such situations by specifying a special type of a figure (*CompositeFigure*), which should be interpreted as a container for other figures. Composite figures should not contain other composite figures. Permitting the construction of complicated hierarchical structures could lead to ambiguities at the stage of interpretation, while not providing considerably higher expressiveness of the ontology. Figure 2.3 shows an example of a description of subfigures.

Besides the specification of the type, the top-level description of figures consists of other attributes. To achieve this level of description, we use some of the properties from the Dublin



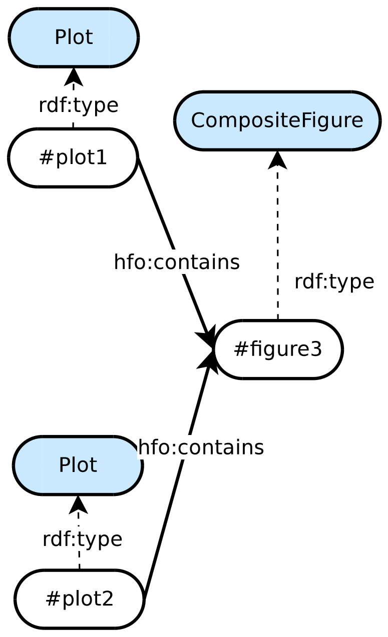


Figure 2.3: Annotation of subfigures

Core ontology together with specially defined HFO predicates. The most visible of the annotation properties is the caption of a figure. Captions summarise the content of a figure and thus contain relevant keywords which can be used in the implementation of figure search. When annotating figures with their captions, HFO uses the Dublin Core title field. Many figures contain a short textual summary appearing at the top of the image. This description plays a very similar role to this of a caption. However, descriptions are part of a figure and as such, should be described separately from a caption which is part of the publication, but describes a figure. Figures can be annotated with their internal descriptions using the *dc:description* predicate.

The name of a figure is another property which describes a figure in the text of the enclosing publication. A name has the form of a short string which identifies the figure in every descriptive context. In most of the cases, the name appears also at the beginning of the caption. However, the name at the beginning of a caption can have a different spelling or capitalisation from the name referred from the text (e.g. FIG. 1 vs Figure 1). Names can be specified by using the string-valued *dc:identifier* attribute.

When a figure is referred from the text of a publication, the surrounding text typically contains useful keywords, which describe the figure's content. Every figure can be referred multiple times. Every reference can be represented by providing a string-valued *hfo:hasTextReference* predicate. We leave for later specification what exactly should be understood as a text reference. In general, a reference is a string within the text of a publication, where a figure name is cited. However, the length of the text can be different. A reference can contain one or two sentences surrounding the figure name. It can also be measured in characters. We do not describe the location of text references inside the text of a publication. In addition to the aforementioned metadata fields, we defined a somehow redundant *hfo:hasFulltext* attribute. The value of this attribute has to be a string literal and it should consist of a space-delimited list of words appearing inside a figure. The order of the words is not significant.

Besides the typical description summarised before, figures are annotated with the HEP concepts represented by the figure, e.g. the name given to a specific type of simulation or an elementary particle. Typically, these attributes would be the same for a figure and for the publication from which the figure originates. In particular, entities of various types in HFO are linked with concepts from the HEP taxonomy [39] using the *dc:subject* property.

Figure 2.4 shows an image which comes from a publication stored in INSPIRE[7], which we will use as an example of annotation along this chapter. Figure 2.5 shows the top-level annotation. The central part of it consists of the #figure entity. The figure is connected with the publication from which it has been extracted using the *hfo:extractedFrom* property. It is worth noting that the same figure can be potentially extracted from many different publications. However, at this level of the description we consider a single figure entity to correspond to a particular figure appearing in a particular publication. If the content of two figures is the

[7]`http://inspirehep.net/record/1238420`

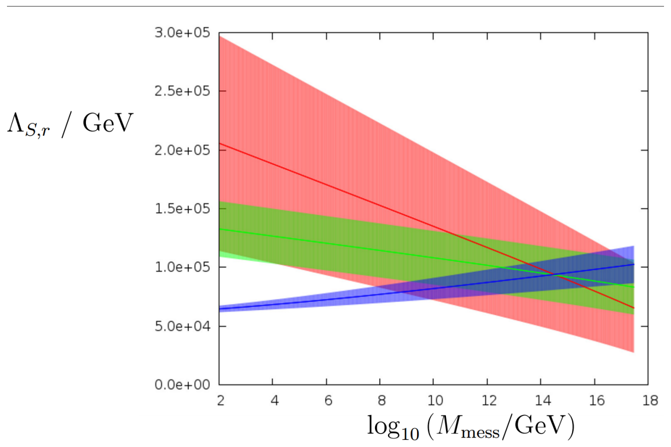


Figure 2.4: The figure used as an example for the annotation.

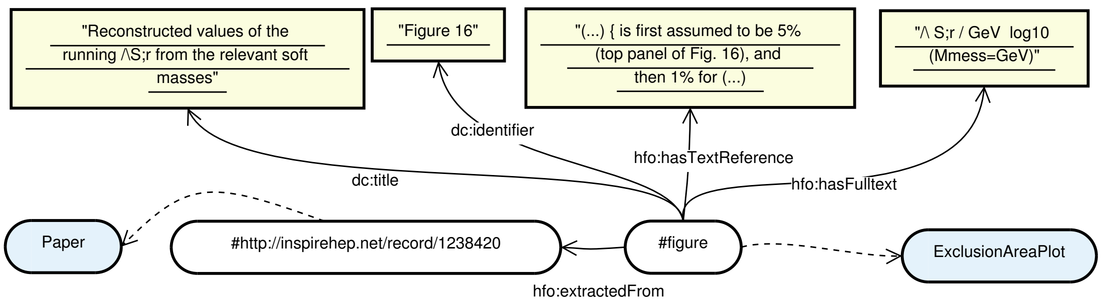

"Reconstructed values of the running /\S;r from the relevant soft masses"
"Figure 16"
"(...) { is first assumed to be 5% (top panel of Fig. 16), and then 1% for (...)
"/\ S;r / GeV log10 (Mmess=GeV)"
dc:identifier
hfo:hasTextReference
dc:title
hfo:hasFulltext
Paper
#http://inspirehep.net/record/1238420
#figure
ExclusionAreaPlot
hfo:extractedFrom

Figure 2.5: The annotation of the main figure entity

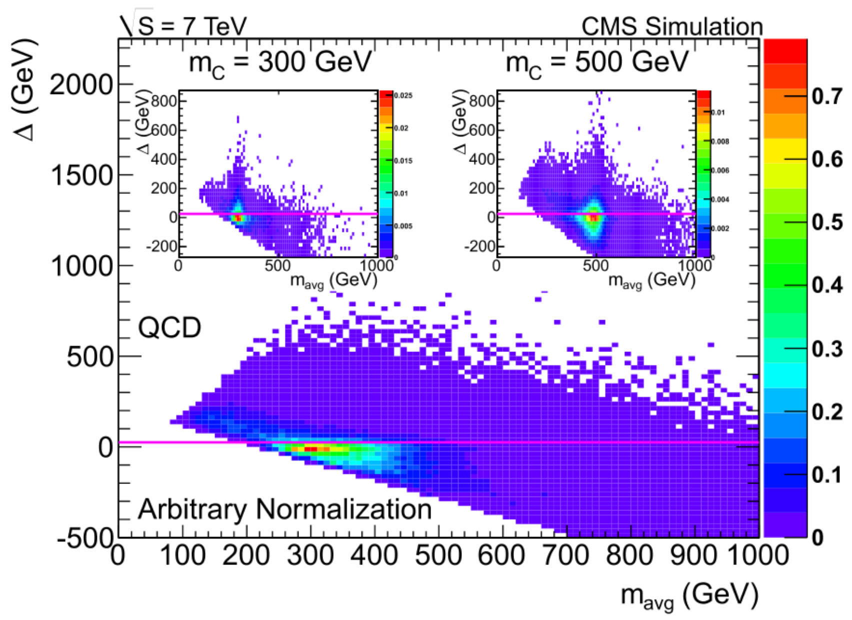


Figure 2.6: Example of a plot containing multiple coordinate systems.

same or similar, users of the ontology can connect the related entities with additional predicates. The presented example contains also several properties referring to a figure within the context of a concrete publication: the name by which the figure is referred from within the text of the publication (*dc:identifier*); the caption which the figure has in the context of the publication (we use the Dublin Core *dc:title* predicate) and references from the text of a publication (*hfo:hasTextRefernce*).

2.3.2 Description of Coordinate Systems

Coordinate systems are amongst the most prominent parts of plots. Typically, there is a single coordinate system appearing within a single plot. However, sometimes we might encounter exceptions from this rule. A plot containing multiple coordinate systems is different from a figure containing multiple plots. Figure 2.6 shows a plot containing many coordinate systems. The difference between the two cases is not obvious. In our considerations, we consider a plot to contain multiple coordinate systems if the rectangles spanned by those systems contain a non-empty intersection.

Usually coordinate systems contain two axes, but three-dimensional plots are not unusual. Figure 2.7 shows the most important elements used to describe axes. The left part of the figure shows the taxonomy of concepts and the right side shows possible relations between their

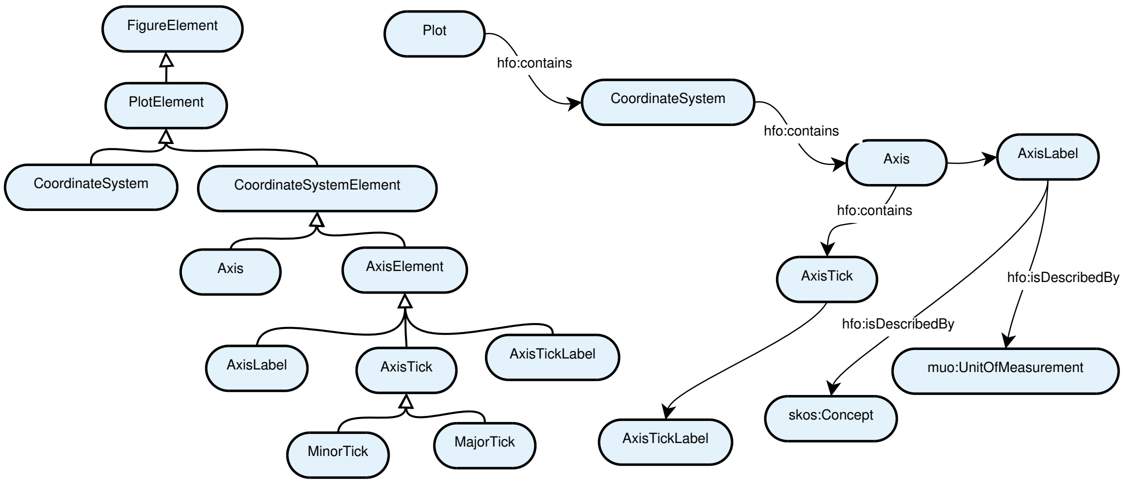


Figure 2.7: The taxonomy of figure elements and a graph of relations between concepts

instances. A figure can be connected using the *hfo:contains* relation with a coordinate system, which in turn can contain axes. Axes contain axis ticks and can be described by axis labels.

Axis ticks serve as a reference during the interpretation of the data. They link positions on axes with concrete numerical values. In a plot, ticks are represented by small lines intersecting axes. HFO describes ticks as points having only one coordinate - distance from the origin of the coordinates system. Ticks can be of two different types: major and minor. Typically, major ticks are assigned a label encoding the corresponding numerical value. Sometimes such a label is attached only to first ticks, indicating the pattern according to which the attachment to the remaining ticks could continue. Minor ticks typically do not have labels. Their corresponding numerical values can be derived in a non-ambiguous manner from their location relative to nearest annotated ticks. HFO represents both types of ticks using special classes inheriting from *AxisTick*. *AxisTick* instances must have an *hfo:atPosition* attribute, which indicates the location of a tick relative to the axis. The position is represented as a floating-point literal indicating the fraction of the total length of an axis. The tick is located at the specified distance from the axis origin. Every axis has a defined abstract direction which does not have to correspond to the natural direction of a plot. If the number assigned with *hfo:atPosition* is positive, the tick is located on the positive side of the beginning of a coordinate system. If the number is negative, it is located on the negative side.

At the level of the general description, axis labels and tick labels could be represented as string literals. However, the material description adds more attributes to those objects. HFO encodes axes and tick labels using individuals of a specialised type, which can be linked with the corresponding tick or label using the *hfo:hasLabel* predicate. The text representation of a label can be linked with the label entity using the Dublin Core *title* attribute.

Typically, axes express a measurement of a certain quantity or particle using a specified unit. Units and particles are linked with axes using *hfo:isDescribedBy*.

The *hfo:contains* predicate plays a central role in the ontology of figures. It allows expressing the aggregation of smaller objects inside a larger and more general one. Individuals connected using the *hfo:contains* attribute form a tree. It is not enforced by the schema, but circular dependencies are not allowed. The relation of containing is transitive. If a coordinate system contains an axis, the same axis is contained by the plot and also, in the case of the main figure containing subfigures, by the main figure. *Hfo:contains* also serves as a super-class for several more specific relations. Such a construct is used to express the fact that one object contains another in a special role. This is especially useful, when we need to distinguish between two or more instances of the same type. More examples can be found in the section about describing the data description.

The *hfo:contains* relation can transport *hfo:isDescribedBy* relations into the container entities. If A *hfo:contains* B and B *hfo:isDescribedBy* C, then also A *hfo:isDescribedBy* C. Such basic inference rules belong to the main ontology because their resolution is simple and they play an important role when formulating concise queries operating on our knowledge base.

2.3.3 Description of Units

Measurements in experimental sciences, in particular in physics, are related to a certain unit of measurement. For instance, the information that the length of a certain object is 5 is insufficient. The numerical value can refer to 5 meter, 5 feet, 5 inches and so on. Moreover, the unit can be equipped with a prefix modifying the value. 5km is for example equivalent to 5000 meters. There exist strict rules to compare and convert measurements between different units of measurement. Sometimes units of measurement are composed of other units. The velocity can be for example measured in kilometres per hour (kmph). This unit is related to the kilometre and the hour. Kilometre in turn can be constructed from meter by adding a prefix.

There are different approaches to representing units. The UO ontology[8] defines different units; however it does not provide means of expressing semantic connections between them. For this reason, similarly to the work described in [48], we decided to use the Measurable Units Ontology (MUO).[9] MUO consists of two parts. The first part describes abstract classes that allow the expression of dependencies between different units (e.g. *BaseUnit*, *DerivedUnit*, *MetricUnit*, *SimpleDerivedUnit*). The second part consists of a set of individuals representing actual units. Such a detailed description of units gives us a much higher flexibility when performing searches. For example, one can find all the individuals that have a quantity measured in a unit related to meters. In addition, MUO provides facilities for describing the measured

[8]`http://obofoundry.org/cgi-bin/detail.cgi?unit`
[9]`http://idi.fundacionctic.org/muo/`

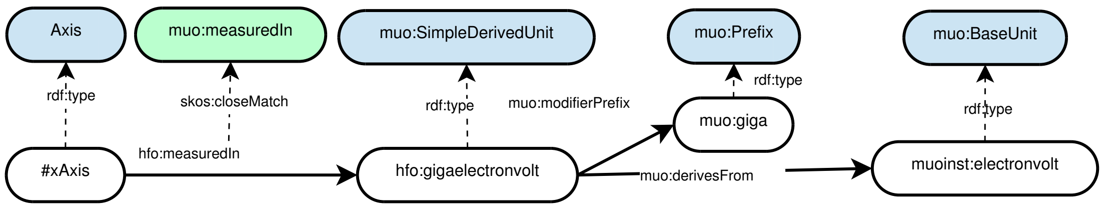


Figure 2.8: Relation between an axis and the corresponding unit

quantities. It provides for example means of expressing the fact that time can be measured in seconds. This type of relations can be useful, when interpreting user queries, by adding the additional level of indirectness. RDF does not provide a standard method of linking numerical literals with their units [48].

For example, one can annotate an individual with a literal describing the magnitude of a quantity and a separate attribute specifying the unit. Another approach can involve encoding values as string literals of a form similar to “666 GeV”. Both solutions loose part of the meaning. The approach of MUO (and also our ontology) involves defining a fully-fledged RDF instance describing the measurement result. Such an instance can be represented by a RDF blank node and should be an instance of a subclass of *muo:QualityValue*. Such an individual can be assigned a numerical value using the *muo:numericalValue* property. It can be also assigned the unit using *muo:measuredIn*.

However, including the concrete measurement values is not very frequent when describing figures in HFO. The method of describing axes allows the data to abstract from concrete numerical values and only links to the abstract unit. Figure 2.8 shows an example of the usage of MUO to describe an axis. We can see that even if the unit related to the axis is gigaelectronvolt, there exists also an indirect link to the basic concept of electronvolt which is an individual belonging to the set provided with MUO.

2.3.4 Description of Data Lines and Areas

Before we delve into the details of the description of the data of plots, we must describe the basic design decisions which we have made. First of all, we describe only plots which map real numbers into real numbers. This assumption is consistent with a two-dimensional nature of the medium used to present plots (a sheet of paper or a computer screen). The second decision involves the way of representing the data. HFO describes plot’s content in terms of graphical primitives rather than trying to be more general and describe mathematical formulae behind a plot (which might not exist).

Figure 2.9 provides an overview of the entities defined for describing data series. Every data series is described in a framework of a coordinates system. In particular, it is related to a pair of axes. One of them is marked as argument axis and the second as the value axis. Besides the

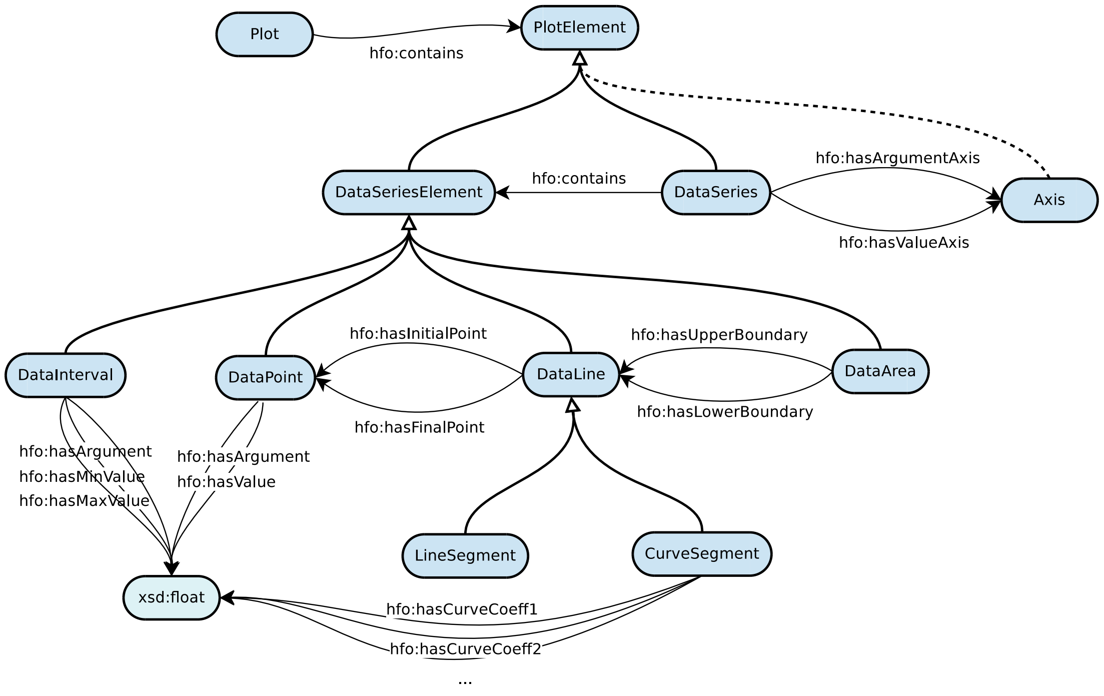


Figure 2.9: The most important entities describing data of a plot

relation with two axes, every *DataSeries* contains a number of entities of the *DataSeriesElement* type, which encode fragments of graphics. There are four major types of *DataSeries* elements:

DataPoint type describes a single point in the 2-dimensional real space; *DataInterval* type to describe mappings between a single real value to an interval of real values; *DataLine* type to describe continuous line segments connecting two data points; and *DataArea* type to describe 2-Dimensional areas. A *DataSeries* containing only *DataPoints* with non-repetitive argument values is called a discrete function series. *DataSeries* containing only *DataLines* whose domains have intersection of measure 0 are called function data series. There are two types of *DataLines*: straight line intervals and curves. *DataAreas* are limited by 2 *DataLines*. They are used to describe area plots - continuous mapping into multiple continuous values. By being allowed to mix different types of *DataSeriesElements*, users of the ontology are provided with high flexibility of describing different types of data series (discrete, continuous, single-valued, multi-valued).

DataSeriesElements are instances of 2-dimensional graphical objects and as such, can be linked with appropriate classes in some ontology describing shapes, for instance the one described in [37] or with concepts from SIO. This in turn opens the possibility of using the HFO description with an arbitrary type of graphical reasoning systems.

Describing geometrical shapes requires providing the coordinates of their points. In the case of semantic description of figures, we wanted to describe data lines in the most general possible way, abstracting from the concrete encoding of the graphs as files. The most obvious approach would involve using the axes with values on them. However, in the case of our main application domain (automatically extracted HEP figures from publications), this approach would lead to difficulties. We cannot assume that the description of a particular axis will not change as a result of a future re-extraction of a document or a manual correction of an extraction error. For this reason we decided to use the features of the axes which were least likely to change, namely the location of the axis line itself. Any two non-parallel lines in the 2-dimensional real space define a coordinates system. When describing points of a plot, axes are mapped to the orthonormal vectors attached at the origin of the coordinates system. This means that we encode all the points using the fraction of the axis length as a reference value, i.e. the point where axes intersects has (0, 0) coordinates. The end of the x-axis is encoded as (1,0) and so on. Most of the data points and shapes located in the area of a plot have both their coordinates in the range between -1 and 1, however it is possible to encounter shapes exceeding the boundary of coordinate systems. Such a representation adds one more level of indirection when reading actual numerical values from a graph, but it makes the description invariant to most of the modifications of the descriptions of coordinate systems.

Figure 2.10 shows part of the annotation of the data appearing inside Figure 2.4, which had already been annotated with external metadata in the previous example. When describing the data lines in the terms of the general ontology, we are agnostic to the existence of axis ticks and to labels associated with them. Ticks and their numerical values should be considered as reference during the interpretation of the data. This however lies outside of the scope of this thesis.

In the case of a description of the blue area from the plot, the entire data series consists of a single solid area. The area is delimited by two lines. The data lines are connected with the area using *hfo:hasLowerBoundary* and *hfo:hasUpperBoundary* properties. The reasoner could use those properties to derive that the data area contains the aforementioned lines and that the entire figure in turn contains those lines. The data area of our example can be annotated with an additional property indicating the predominant colour (Figure 2.11).

2.3.5 The Material Description

The material description of figures plays a supplementary role in their description. However, it is useful in every place where the image representation has to be related to the semantic content of the figure. It is internally used during the figure extraction and during the automatic semantic analysis. It can also be used to allow users to manually correct errors produced by the automatic procedures. Material description can form a basis for the development of tools

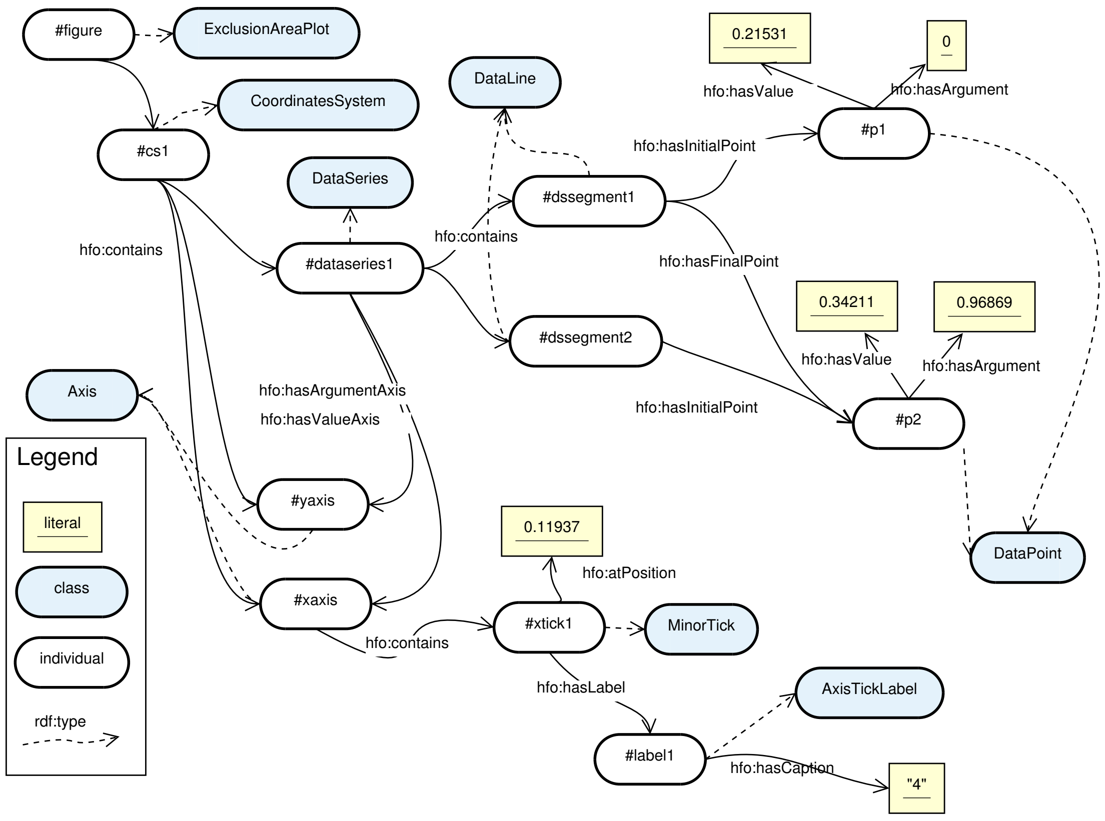


Figure 2.10: Part of the annotation of an example figure

that enable the interaction with the content of figures.

All the information that is essential for the interpretation of figure content can be encoded using the concepts of the central part of the ontology. The material annotation links elements of the abstract description of a figure with its particular representation as graphics. The annotation of the same figure encoded in two graphical files having different canvas sizes or intrinsic coordinates systems will differ. However, the description of the same figure using different file formats, sharing the coordinates systems, will be identical.

Many entities from the central ontology are represented in the original graphics as a sum of certain number of graphical entities. For example, inside the graphical file axes are represented by lines, legends and captions can be enclosed in rectangles. Those graphical primitives can be described using subclasses of the graphical entity class present in Semantic Science Integrated Ontology[10]. As the material description is dependent on the intrinsic coordinates system of the representation as a file, those graphical primitives can be described in the same coordinates

[10]`http://semanticscience.org`

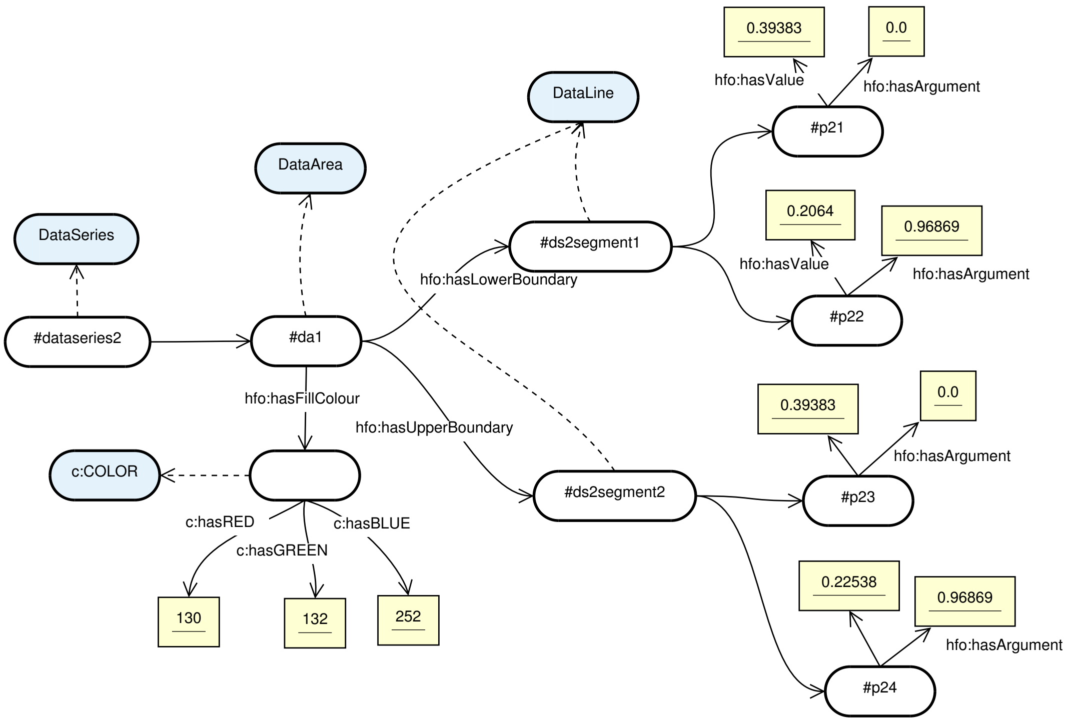


Figure 2.11: The description of the blue area of the example plot

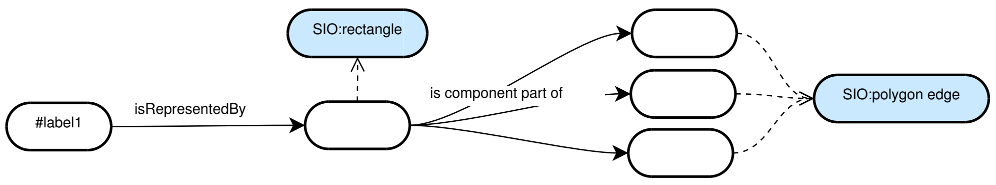


Figure 2.12: An example of the material description of a tick label

system. The link between concepts from the central ontology and the graphical primitives can be established using the *hfo:isRepresentedBy* property (see Figure 2.12).

The material description should be used only with those elements of a figure whose meaning does not reveal their position in the original medium. It should be avoided in the places where position is encoded in an indirect manner. For example, data lines are encoded using the coordinates in an abstract coordinates system. The material description provides a link between the abstract coordinates system and the canvas coordinates system. This link can be used to translate the coordinates of data into the canvas system.

2.4 Summary

This chapter has presented the ontology allowing a description of HEP figures. The provided vocabulary allows describing all types of figures at a general level. It also contains the vocabulary which allows the description of plots (the most common type of figures) with most of the detail of their internal semantics.

HFO enables the creation of knowledge bases for storing the semantic description of scientific graphs and forms a basis that allows the implementation of semantics-aware searching. As such, HFO is one of the main pillars of the system enabling the management of scientific figures. HFO can also be used to establish a common vocabulary shared between different systems which process figures, increasing the semantic interoperability and advancing the vision of the Semantic Web. Many figure types present in HFO and typical for HEP appear also in other disciplines and thus the present results can be reused in a different content. Others can be added in a compatible manner.

Additional work is required to enable the description of different types of figures typical in HEP publications (like different types of scientific diagrams). Figures are only one type of objects that could be annotated. In the future, HFO could be extended to allow the annotation of those other types of objects. Having a consistent description of different types of scholarly objects would facilitate the construction of uniform methods for comparing them and searching.

The vocabulary described in this chapter is used as a metadata model for the results generated by the prototype systems described in the two following chapters.

Chapter 3

Extraction of Figures from Scholarly Publications

3.1 Introduction

The previous chapter described a model allowing the description of the semantics of HEP figures. This chapter together with Chapter 4 describe a method that enables the automatic generation of such annotation based only on the file containing a figure. At this place we present a method of extracting figures from scholarly publications encoded in Portable Document Format (PDF) documents. Thanks to the method proposed in this chapter, it is possible to generate part of the metadata/annotations according to the general ontology proposed in the previous chapter. The metadata produced at the stage of extracting figures, does not directly describe the content of a figure. It relates figure to the context of the publication in which it has been included. The extraction and the annotation of the content of figures are described in the following chapter.

The additional results of the extraction algorithm consist of raster images of a figure and also vector graphics. These data are used during the further processing.

The PDF format has been chosen as the input of the algorithm because it is a de facto standard in scientific communication. In the case of HEP, mathematics and other exact sciences, the majority of publications are prepared using the LaTeXdocument formatting system and later they are compiled to a PDF file. The electronic versions of publications from major scientific journals are also provided in PDF format. The internal structure of PDF files does not always reveal the location of graphics. In some cases images are included as external entities and easily distinguishable from the rest of documents content, but other times they are mixed with the rest of the content. Therefore, in order not to miss any figure, the low-level structure of a PDF must be analysed. The work described in this chapter focuses on the area of High Energy Physics. However, with minor variations, described methods could be applicable in the case of a different area of knowledge.

The remainder of the chapter is organised as follows: Section 3.2 presents the state of the art in the area of PDF document analysis and figure extraction; Section 3.3 describes every step of the extraction method in detail; Section 3.4 presents the results of the evaluation of the presented method on a test-bed of HEP articles; Finally, the chapter ends with some conclusions and outlook on future work.

3.2 Related Work

Over years of development of Digital Libraries and document processing, researchers elaborated several methods of automatically extracting and processing graphics appearing in PDF documents. These methods can be divided into two groups by considering the properties of the processed content. The attempts of the first category deal with PDF documents in general, not making any assumptions about the content of encoded graphics or the document type. The methods from the second group are more specific to figures from scientific publications. The presented approach belongs to the second group.

General tools include command line programs like *PDF-Images*[1] and web-based applications like *pdftoword*[2]. These solutions are useful in the general case of documents, but all suffer from the same difficulties when processing scientific publications: Graphics that are recognised by such tools have to be marked as graphics inside PDF documents. This is the case with raster graphics and some other internally stored types of objects. In the case of scholarly documents, most graphics are constructed internally using PDF primitives and thus cannot be correctly processed by tools from the first group. Moreover, general tools do not have the necessary knowledge to produce metadata describing the extracted content.

Most of existing works aiming at the retrieval and analysis of figures use the rasterised graphical representation of source documents as its basis. Browuer et al. [9, 31] describe a method of detecting plots by means of wavelet analysis. In their work they focus on the extraction of data points from identified figures. In particular, they address the challenge of correctly identifying overlapping points of data in plots. This is an example of a problem which would not manifest itself often in the case of vector graphics, which is the scenario proposed in our extraction method. Vector graphics preserve much more information about the documents content than simple values of pixel colours. In particular, vector graphics describe overlapping objects separately. Raster methods are also much more prone to additional errors being introduced during the recognition/extraction phase. The methods described in this chapter could be used along with the method of Kataria [30] for the case of documents being output of a digitisation process.

Liu et al. [34] present a page box-cutting algorithm for the extraction of tables from PDF

[1]http://sourceforge.net/projects/pdf-images/ (last access: 20.02.2013)
[2]http://www.pdftoword.com/ (last access: 20.02.2013)

documents. Their approach is not directly applicable but their ideas of geometrical clustering of PDF primitives are similar to the ones proposed in our work. However, our experiments with their implementation and a testing set of HEP publications have shown that the heuristics used in their work cannot be directly applied to the case of HEP, showing the need for an adapted approach, even in the case of tables.

A different category of work, not directly related to graphics extraction, but useful when designing graph extraction algorithms, has been devoted to the analysis of graph usage in scientific publications. The results presented by Cleveland et al. [12] describe a more general case than publications of High Energy Physics. Even if the data presented in the work came from scientific publications before 1984, included observations, as for example typical sizes of graphs, were useful with respect to general properties of figures and were taken into account when adjusting the initial parameters of the presented algorithm.

Finally, there exist attempts to extract layout information from PDF documents. The knowledge of page layout is useful to distinguish completely independent parts of the content. The approach of layout and content extraction presented by Chao et al. [11] is the closest to the one we propose in this chapter. The difference lies in the fact that we are focusing on the extraction of plots and figures from scientific documents, which usually follow stricter conventions. Therefore, we can make more assumptions about their content and extract more precise data. For instance, our method emphasises the role of detected captions and permits them to modify the way in which graphics are treated. We also extract portions of information that are difficult to be extracted using more general methods, such as captions of figures.

3.3 The Method

PDF files have a complex internal structure that allows embedding various external objects and to include various types of metadata. However, the central part of every PDF file consists of a visual description of the subsequent pages. The imaging model of PDF uses a language based on a subset of the PostScript language. PostScript is a complete programming language containing instructions (called also operators) which allow rendering text and images on a virtual canvas. The canvas can correspond to a computer screen or to another, possibly virtual, device used to visualise the file. The subset of PostScript, which was used to describe content of PDFs had been stripped from all the flow control operations (like loops and conditional executions), which makes it much simpler to interpret than the original PostScript. Additionally, the state of the renderer is not preserved between subsequent pages, making their interpretation independent.

In order to avoid many technical details, which are irrelevant in this context, we will consider a PDF document as a sequence of operators (also called the content stream). Every operator can trigger a modification of the graphical state of the PDF interpreter, which might be drawing a graphical primitive, rendering an external attached object or modifying the position of the graphical pointer[3] or a transformation matrix[4]. The outcome of an atomic operation encoded in the content stream depends not only on parameters of the operation but also on the way previous operators modified the state of the interpreter. Such design makes a PDF file easy to render but not necessarily easy to analyse.

Figure 3.1 provides an overview of the proposed extraction method. At the very first stage, the document is preprocessed and operators are extracted (see Section 3.3.1). Later, graphical and textual operators are clustered using different criteria (see Sections 3.3.3 and 3.3.4) and the first round of heuristics rejects regions which cannot be considered figures. In the next phase, the clusters of graphical operators are merged with text operators representing fragments of text to be included inside a figure (see section 3.3.5). The second round of heuristics detects clusters which are unlikely to be figures. Text areas detected by the means of clustering text operations are searched for possible figure captions (see Section 3.3.6). Captions are matched with corresponding figure candidates and geometrical properties of captions are used to refine detected graphics. The last step generates data in a format convenient for further processing (see section 3.3.7).

Additionally, it must be noted that another important preprocessing step of the method consists of the layout detection. Section 3.3.2 discusses an algorithm segmenting pages into layout elements called page divisions. This considerably improves the accuracy of the extraction method because elements from different page divisions can no longer be considered belonging to the same cluster (and subsequently figure). This allows applying the method separately to different columns of a document page.

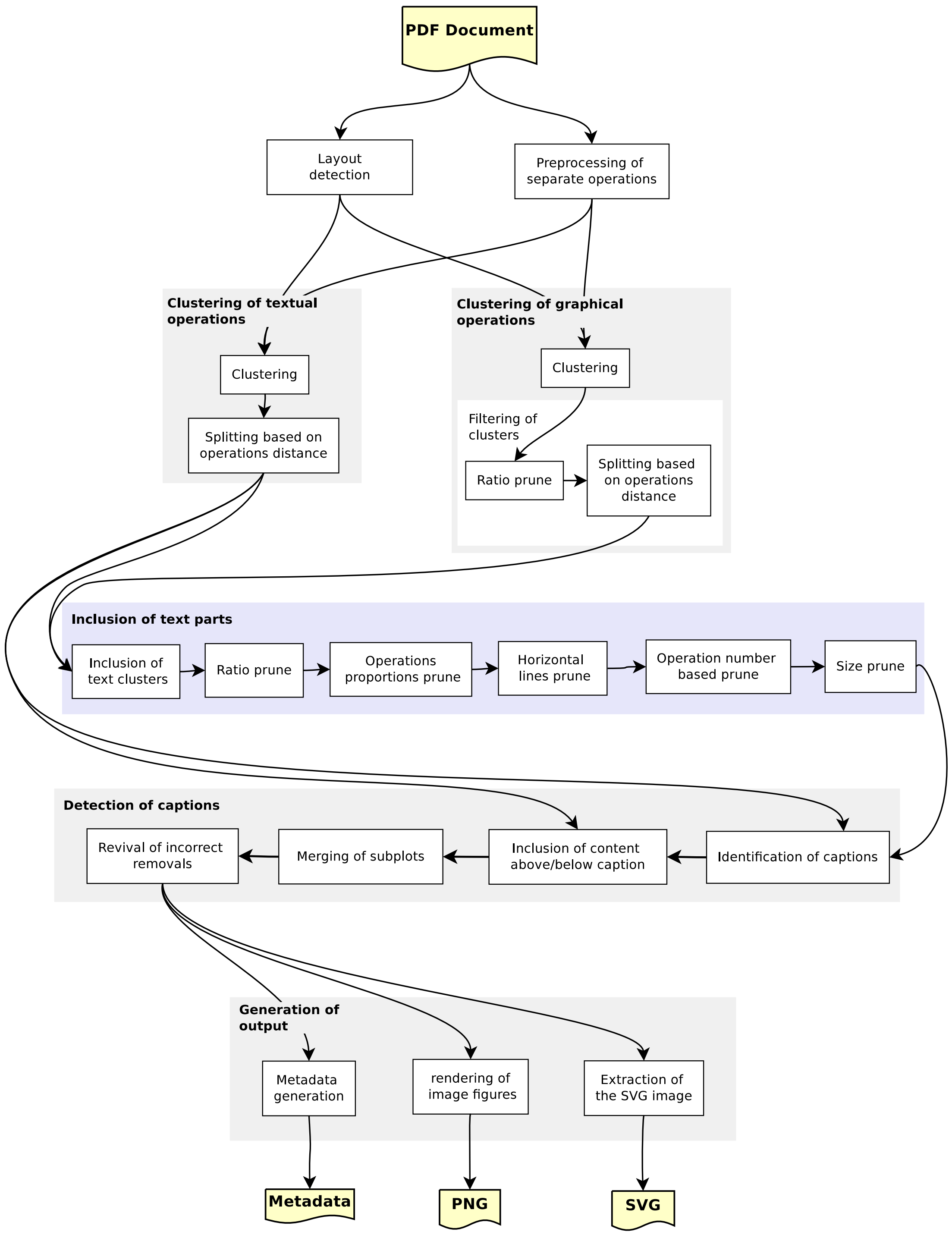
3.3.1 Preprocessing of Operators

The proposed algorithm considers only certain properties of a PDF operator rather than trying to completely understand its effect. Considered properties consist of the operators type (textual, graphical or transformation), the region of the page where the operator produces output and, in the case of textual operations, the string representation of the result. For simplicity, we suppress the notion of coordinate system transformation, inherent for the PDF rendering, and describe all operators in a single coordinate system of a virtual 2-dimensional canvas where operations take effect. Transformation operators (those not producing any visible output but only modifying the interpreter's state) are assigned an empty operation region as they do not modify the result directly but affect subsequent operations.

In our implementation, an existing PDF rendering library has been used to determine

[3]At every moment of the execution of a PostScript program, the interpreter maintains a number of variables. Some of them encode current positions within the rendering canvas. Such positions are used to locate the subsequent character or to define the starting point of the subsequent graphical primitive.

[4]Transformation matrices are encoded inside the interpreters state. If an operator requires arguments indicating coordinates, these matrices are used to translate the provided coordinates to the coordinate system of the canvas.



PDF Document
Layout detection
Preprocessing of separate operations
Clustering of textual operations
Clustering
Splitting based on operations distance
Clustering of graphical operations
Clustering
Filtering of clusters
Ratio prune
Splitting based on operations distance
Inclusion of text parts
Inclusion of text clusters
Ratio prune
Operations proportions prune
Horizontal lines prune
Operation number based prune
Size prune
Detection of captions
Revival of incorrect removals
Merging of subplots
Inclusion of content above/below caption
Identification of captions
Generation of output
Metadata generation
rendering of image figures
Extraction of the SVG image
Metadata
PNG
SVG

Figure 3.1: Overview of the figure extraction method

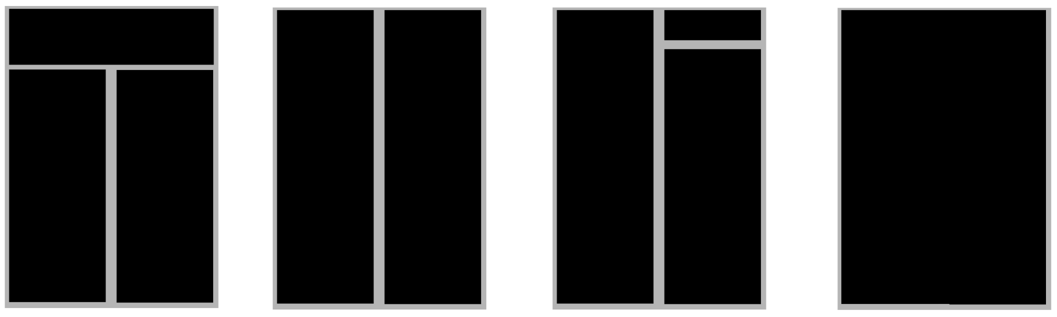


Figure 3.2: Sample page layouts that might appear in a scientific publication. The black colour indicates areas where content is present.

boundaries of operators. Rather than trying to recognise all possible operators, we check the area of the canvas that has been affected by an operation. If the area is empty, we consider the operation to be a transformation. If there exists a non-empty area that has been changed, we check if the operator belongs to a predefined list of textual operators. This list is created based on the PDF specification. If so, the operators argument list is scanned searching for a string and the operation is considered to be textual. An operation that is neither a transformation nor a textual operation is considered to be graphical. It might happen that text is generated using a graphical operator. However, such a situation is unusual. In the case of operators rendering of other operators, which is the case for example when rendering text using type-3 fonts, we consider only the top-level operation.

In most cases, separate operations are not equivalent to logical entities considered by a human reader. Graphical operators are usually responsible for displaying lines or curve segments while humans think in terms of illustrations, data lines and so on. Similarly, in the case of text, operators do not have to represent complete or separate words or paragraphs. They usually render parts of words and sometimes parts of more than one word.

The only assumption we make about the relation between operators and logical entities is that a single operator does not trigger rendering of elements from different detected entities (figures, captions). This is usually true because logical entities tend to be separated by a modification of the context (There is a distance between text paragraphs or an empty space between curves).

3.3.2 Detection of the Page Layout

In this section we discuss how to detect the page layout, an issue which is essential for the detection of figures. Figure 3.2 depicts several possibilities of organising content on the page. Detecting separate logical areas of a page is a preliminary stage improving the quality of the results from the subsequent stages of the clustering algorithm described in the following sections. Being able to identify how the document page is divided into columns enables us to execute the clustering within every column separately.

Intuitively it is very obvious, what can be understood as a page layout, although in order to provide a method of calculating such, we need a more formal definition, which we provide below. By the layout of a page, we understand a special division of page into areas called columns. Each area is a sum of rectangles, the division into areas must satisfy conditions summarised in Definition 3.3.1.

Definition 3.3.1. Let P be a rectangle representing the page. The set D consisting of sets of subareas of a page is called a page division if and only if

$$\bigcup_{Q \in D} Q = P \tag{3.3.1}$$

$$\forall x, y \in D, x \cap y = \emptyset \tag{3.3.2}$$

$$\forall Q \in D, Q \neq \emptyset \tag{3.3.3}$$

$$\forall Q \in D, \exists R = \{x : x \textit{ is a rectangle}\}, Q = \bigcup_{x \in R} x \tag{3.3.4}$$

Every element of a division is called a page area.

In order to be considered a page layout, borders of areas from the division must not intersect the content of the page. Definition 3.3.1 does not guarantee that the layout is unique. A single page might be assigned different divisions satisfying the definition. Additionally, not all page layouts are interesting from the point of view of figures detection. The segmentation algorithm calculates one of such divisions, imposing additional constraints on the detected areas. The layout-calculation procedure utilises the notion of separators, introduced by Definition 3.3.2.

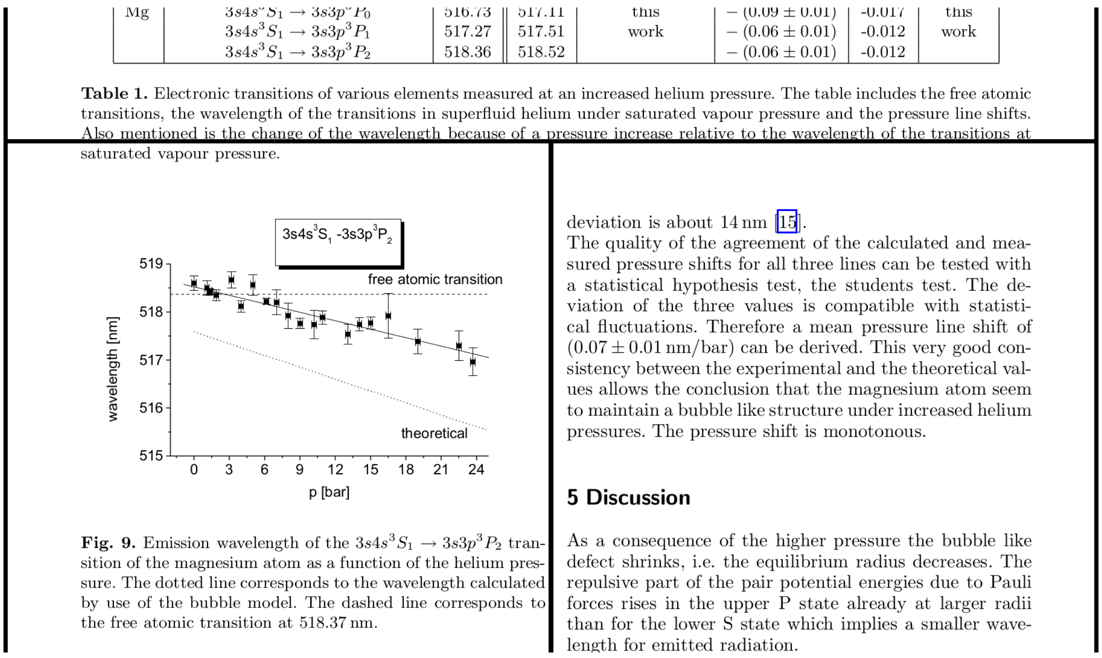
Definition 3.3.2. A vertical (or horizontal) line inside a page or on its borders is called a separator if its horizontal (vertical) distance from the page content is larger than a given value.

The algorithm consists of two stages. First, the vertical separators of a sufficient length are detected and used to divide the page into disjoint rectangular areas. Each area is delimited by two vertical lines each of which forms a consistent interval inside of one of the detected vertical separators. At this stage, horizontal separators are completely ignored. Figure 3.3 shows a fragment of a publication page processed by the first stage of the layout-detection. The upper horizontal edge of one of the areas lies too close too close to two text lines. With the constant of the Definition 3.3.2 chosen to be sufficiently large, this edge would not be a horizontal separator and thus the generated division of the page would require additional processing to become a valid page layout. The second stage of the algorithm transforms the previously detected rectangles into a valid page layout by splitting rectangles into smaller parts and by joining appropriate rectangles to form a single area.

Algorithm 1 shows the pseudo-code of the detection of vertical separators. The input of the algorithm consists of the image of the publication page. The output is a list of vertical

Algorithm 1 Detecting vertical separators

```
1: **Input** : $the\ page\ image$
2: **Output** : $vertical\ separators\ of\ the\ input\ page$
3: $List < Pair < int, List < int >> separators \leftarrow \emptyset$
4: $int\ max\_weight \leftarrow 0$
5: $boolean\ maximising \leftarrow false$
6: for all $x = min_x \ldots \max_x$ do
7:     $empty_b \leftarrow 0, current\_eval \leftarrow 0$
8:     $empty\_areas \leftarrow List()$
9:     for all $y = 0 \ldots page\_height$ do
10:        if $point\ at\ (x, y)\ is\ not\ blank$ then
11:           if $y - empty_b - 1 > height_{min}$ then
12:              $empty\_areas.append(empty_b)$
13:              $empty\_areas.append(y == page\_height?y : y - 1)$
14:              $current\_eval \leftarrow current\_eval + y - empty_b$
15:           end if
16:           $empty_b \leftarrow y + 1$
17:        end if
18:     end for
        {We have already processed the entire column. Comparing with adjacent already processed columns}
19:     if $max\_weight < current\_eval$ then
20:        $max\_weight \leftarrow current\_eval$
21:        $max\_separators \leftarrow empty\_areas$
22:        $max_x \leftarrow x$
23:     end if
24:     if $maximising$ then
25:        if $empty\_areas = \emptyset$ then
26:           $separators.add(< max_x, max\_separators >)$
27:           $maximising \leftarrow false, max\_weight \leftarrow 0$
28:        end if
29:     else
30:        $maximising \leftarrow empty\_areas \neq \emptyset$
31:     end if
32: end for
33: return $separators$
```



Mg	$3s4s^3S_1 \rightarrow 3s3p^3P_0$	516.73	517.11	this	$-(0.09 \pm 0.01)$	-0.017	this
	$3s4s^3S_1 \rightarrow 3s3p^3P_1$	517.27	517.51	work	$-(0.06 \pm 0.01)$	-0.012	work
	$3s4s^3S_1 \rightarrow 3s3p^3P_2$	518.36	518.52		$-(0.06 \pm 0.01)$	-0.012	

Table 1. Electronic transitions of various elements measured at an increased helium pressure. The table includes the free atomic transitions, the wavelength of the transitions in superfluid helium under saturated vapour pressure and the pressure line shifts. Also mentioned is the change of the wavelength because of a pressure increase relative to the wavelength of the transitions at saturated vapour pressure.

Fig. 9. Emission wavelength of the $3s4s^3S_1 \rightarrow 3s3p^3P_2$ transition of the magnesium atom as a function of the helium pressure. The dotted line corresponds to the wavelength calculated by use of the bubble model. The dashed line corresponds to the free atomic transition at 518.37 nm.

deviation is about 14 nm [15].
The quality of the agreement of the calculated and measured pressure shifts for all three lines can be tested with a statistical hypothesis test, the students test. The deviation of the three values is compatible with statistical fluctuations. Therefore a mean pressure line shift of $(0.07 \pm 0.01\,\text{nm/bar})$ can be derived. This very good consistency between the experimental and the theoretical values allows the conclusion that the magnesium atom seem to maintain a bubble like structure under increased helium pressures. The pressure shift is monotonous.

5 Discussion

As a consequence of the higher pressure the bubble like defect shrinks, i.e. the equilibrium radius decreases. The repulsive part of the pair potential energies due to Pauli forces rises in the upper P state already at larger radii than for the lower S state which implies a smaller wavelength for emitted radiation.

Figure 3.3: An example of intermediate layout-detection results requiring the refinement

separators aggregated by their x-coordinates. Every element of this list consists of two elements: an integer indicating the x-coordinate and the list of y-coordinates describing the separators. The first element of this list indicates the y-coordinate of the beginning of the first separator. The second element is the y-coordinate of the end of the same separator. The third and fourth elements describe the second separator and the same mechanism is used for the remaining separators (if they exist).

The algorithm proceeds according to the sweeping principle [13] known from the computational geometry. The algorithm reads the publication page starting from the left. For every x-coordinate value, a set of corresponding vertical separators is detected (lines 9 — 18). Vertical separators are searched as consistent sequences of blank points. A point is considered blank if all the points in its horizontal surrounding of the radius defined by the constant from Definition 3.3.2 are of the background colour. Not all blank vertical lines can be considered separators. Short empty spaces usually delimit lines of text or different small units of the content. In line 11 we test detected vertical separators for being long enough.

If a separator has been detected in a particular column of a publication page, the adjacent columns also tend to contain similar separators. Lines 19-31 of the algorithm are responsible for electing the longest candidate among the adjacent columns of the page. The maximisation is performed across a set of adjacent columns for which at least one separator exists.

The detected separators are used to create the preliminary division of the page (Similar to the one from the example of Figure 3.3). Similarly to the previous step, separators are considered one by one in the order of increasing x coordinate. At every moment of the execution, the algorithm maintains a division of the page into rectangles. This division corresponds only to the already detected vertical separators. Updating the previously considered division is facilitated by processing separators in a particular well-defined order.

Before presenting the final outcome, the algorithm must refine the previously-calculated division. This happens in the second phase of the execution. All the horizontal borders of the division are then moved along adjacent vertical separators until they become horizontal separators in the sense of Definition 3.3.2. Typically, moving the horizontal borders result in dividing already existing rectangles into smaller ones. If such a situation happens, both newly created parts are assigned to different page layout areas. Sometimes when moving separators is not possible, different areas are combined together, forming a larger one.

3.3.3 Clustering of Graphical Operators

The Clustering Algorithm

The representation of a document as a stream of rectangles allows the calculation of more abstract elements of the document. In our model, every logical entity of the document is equivalent to a set of operators. The set of all operators of the document is divided into disjoint subsets in the process called clustering. Operators are decided to belong to the same cluster based on the position of their boundaries. The criteria for the clustering are based on a simple but important observation: operations forming a logical entity have boundaries lying close to each other. Groups of operations forming different entities are separated by empty spaces.

The clustering of textual operations yields text paragraphs and smaller objects like section headings. However, in the case of graphical operations, we can obtain consistent parts of images, but usually not complete figures yet. Outcomes of the clustering are utilised during the process of figures detection.

Algorithm 2 shows the pseudo-code of the clustering algorithm. The input of the algorithm consists of a set of preprocessed operators annotated with their affected area. The output is a division of the input set into disjoint clusters. Every cluster is assigned a boundary equal to the smallest rectangle containing boundaries of all included operations.

In the first stage of the algorithm (lines 5-19), we organise all input operations in a data structure of forest of trees. Every tree describes a separate cluster of operations. The second stage (lines 20-29) converts the results (clusters) into a more suitable format.

The clustering of operations is based on the relation of their rectangles being close to each other. Definition 3.3.3 formalises the notion of being close, making it useful for the algorithm.

Algorithm 2 Clustering algorithm

```
1: Input : $OperationSet\ input\_operations${set of operators of the same type}
2: Output : $Map < Rectangle, OperationSet >${spatial clusters of operators}
3: $IntervalTree\ t_x \leftarrow IntervalTree()$
4: $IntervalTree\ t_y \leftarrow IntervalTree()$
5: $Map < Operation, Operation > parent \leftarrow Map()$
6: for all $Operation\ op \in input\_operations$ do
7:     $Rectangle\ boundary \leftarrow extendByMargins(op.boundary)$
8:     repeat
9:         $OperationSet\ int\_ops_x \leftarrow t_x.getIntersectingOps(boundary)$
10:        $OperationSet\ int\_ops_y \leftarrow t_y.getIntersectingOps(boundary)$
11:        $OperationSet\ int\_ops \leftarrow int\_ops_x \cap int\_ops_y$
12:        for all $Operation\ int\_op \in int\_ops$ do
13:            $Rectangle\ bd \leftarrow t_x[int\_op] \times t_y[int\_op]$
14:            $boundary \leftarrow smallestEnclosing(bd, boundary)$
15:            $parent[int\_op] \leftarrow op$
16:            $t_x.remove(int\_op), t_y.remove(int\_op)$
17:        end for
18:    until $int\_ops = \emptyset$
19:    $t_x.add(boundary, op), t_y.add(boundary, op)$
20: end for
21: $Map < Rectangle, OperationSet > results \leftarrow Map()$
22: for all $Operation\ op \in input\_operations$ do
23:    $Operation\ root\_ob \leftarrow getRoot(parent, op)$
24:    $Rectangle\ rec \leftarrow t_x[root\_ob] \times t_y[root\_ob]$
25:    if $notresults.has\_key(rec)$ then
26:        $results[rec] \leftarrow List()$
27:    end if
28:    $results[rec].add(op)$
29: end for
30: return $results$
```

Definition 3.3.3. Two rectangles are considered to be located close to each other if they are intersecting after expanding their boundaries in every direction by a margin.

The value by which rectangles should be extended is a parameter of the algorithm and might be different in various situations. In order to detect if rectangles are close to each other, we needed a data structure able to store a set of rectangles. This data structure was required to allow retrieving all stored rectangles that intersect a given one.

We have constructed the necessary structure using an important observation about the operation result areas. In our model all bounding rectangles have their edges parallel to the edges of the reference canvas on which the output of the operators is rendered. This allowed us to reduce our problem from the case of 2-dimensional rectangles to the case of 1-dimensional intervals. We can assume that edges of the rectangular canvas define the coordinates system. It is easy to prove that two rectangles of edges parallel to the axis of the coordinates system intersect only if both their projections in the directions of axis intersect. The projection of a rectangle into an axis is always an interval.

The observation made above has allowed us to build the required 2-dimensional data structure by remembering 2 one-dimensional data structures that allows remembering a number of intervals and for a given interval return the set of intersecting ones. Such a one-dimensional data structure has been provided by interval-trees[20]. Every interval inside the tree has an arbitrary object assigned to it, which in this case is a representation of the PDF operator. This object can be treated as an identifier of the interval. The data structure also implements a dictionary interface, mapping objects to actual intervals.

At the beginning, the algorithm initialises two empty interval trees representing projections on X and Y axis respectively. Those trees store values about projections of the biggest so-far calculated areas rather than about particular operators. Each cluster is represented by the most recently discovered operation belonging to it.

During the algorithm execution, each operator from the input set is considered only once. The order of processing is not important. The processing of a single operator proceeds as follows (the interior of the outermost "for all" loop of the algorithm).

1. Firstly, the boundary of the operation is extended by the width of margins. The spatial data structure described earlier is utilised to retrieve boundaries of all already detected clusters (lines 8-9)

2. The forest of trees representing clusters is updated. The currently processed operation is added without a parent. Roots of all trees representing intersecting clusters (retrieved in the previous step) are attached as children of the new operation.

3. The boundary of the processed operation is extended to become the smallest rectangle containing all boundaries of intersecting clusters and the original boundary. Finally, all intersecting clusters are removed from the spatial data structure.

4. Lines 8-16 of the algorithm are repeated as long as there are areas intersecting the current boundary. In some special cases, more than one iteration may be necessary.

5. Finally, the calculated boundary is inserted into the spatial data structure as a boundary of a new cluster. The currently processed operation is designed to represent the cluster and so, is remembered as a representative of the cluster.

After processing all available operations, the post-processing phase begins. All the trees are transformed into lists. The resulting data structure is a dictionary having boundaries of detected clusters as keys and lists of belonging operations as values. This is achieved in lines 20-29. During the process of retrieving the cluster to which a given operation belongs, we use a technique called path compression, known from the Find&Union data structure [13].

Filtering of Clusters

Graphical areas detected by a simple clustering usually do not directly correspond to figures. The main reason for this is that figures may contain not only graphics, but also portions of text. Moreover, not all graphics present in the document must be part of a figure. For instance, common graphical elements not belonging to a figure include logos of institutions and text separators like lines and boxes; various parts of mathematical formulas usually include graphical operations; and in the case of slides from presentations, the graphical layout should not be considered part of a figure.

The above shows that the clustering algorithm described earlier is not sufficient for the purpose of figures detection and it yields a result set wider than expected. In order to take into account the aforementioned characteristics, precalculated graphical areas are subject to further refinement. This part of the processing is highly domain-dependent as it is based on properties of scientific publications in a particular domain, in this case publications of HEP. In the course of the refinement process, previously computed clusters can be completely discarded, extended with new elements, or some of their parts might be removed. In this subsection we discuss the heuristics applied for rejecting and splitting clusters of graphical operators.

There are two main reasons for rejecting a cluster. The first of them is a size being too small compared to a page size. The second is the figure candidate having its aspect ratio outside a desired interval of values.

The first heuristic is designed to remove small graphical elements appearing for example inside mathematical formulas, but also small logos and other decorations. The second one discards text separators and different parts of mathematical equations, such as a line separating numerator from a denominator inside a fraction. The thresholds used for filtering are provided as configurable properties of the algorithm and their values are assigned experimentally in a way maximising the accuracy of figures detection.

Additionally, the analysis of the order of operations forming the content stream of a PDF document may help to split clusters that were incorrectly joined by Algorithm 2. Parts of the stream corresponding to logical parts of the document usually form a consistent sub-sequence. This observation allows constructing a method for splitting elements that were clustered together incorrectly. We can assign content streams not only to entire PDF documents or pages, but also to every cluster of operations. The clustering algorithm presented in algorithm 2 returns a set of areas with a list of operations assigned to each of them. The content stream of a cluster consists of all operations from such a set ordered in the same manner as in the original content stream of the PDF document. The usage of the original content stream allows us to define a distance in the content stream as follows:

Definition 3.3.4. If o_1 and o_2 are two operations appearing in the content stream of the PDF document, by the distance between these operations we understand the number of textual and graphical operations appearing after the first of them and before the second of them.

In order to detect situations when a figure candidate contains unnecessary parts, the content stream of a figure candidate is read from the first to the last operation. For every two subsequent operations, the distance between them in the sense of the original content stream is calculated. If the value is larger than a given threshold, the content stream is split into two parts which become separate figure candidates. For both candidates, a new boundary is calculated.

This heuristic is especially important in the case of less formal publications such as slides from presentations at conferences. Presentation slides tend to have a certain amount of graphics appearing on every page and not carrying any meaning. Simple geometrical clustering would connect elements of page style with all the rest of the document content. Measuring the distance in the content stream and defining a threshold on the distance facilitates the distinction between the layout and the rest of the page. This technique might be also useful in order to automatically extract the template used for a presentation, although this transcends the scope of this publication.

3.3.4 Clustering of Textual Operators

The same algorithm that is applied to cluster graphical elements can be used to cluster parts of text. Detecting larger logically consistent parts of text is important because they should be treated as single entities during subsequent processing. This comprises inclusion inside a figure candidate (captions of axes, parts of a legend etc...), classification of a text paragraph as a figure caption and so on.

3.3.5 Inclusion of Text Parts

The next step in figures extraction involves the inclusion of lost text parts inside figure candidates. At the stage of operations clustering, only the operations of the same type (graphical or textual) were considered. The results of those initial steps become subsequently the input to the clustering algorithm that will detect relations between previously detected entities. By doing this, we move one level farther in the process of abstracting from operations. Initially we start from basic meaningless operations. Later we detect parts of graphics and text and finally we are able to see the relations between both.

Not all clusters detected at this stage are interesting because some clusters might consist uniquely of text areas. Only those results that include at least one graphical cluster may be subsequently considered figure candidates.

Another round of heuristics allows marking unnecessary intermediate results as deleted. Applied methods are very similar to those described in Section 3.3.3, only thresholds deciding on the rejections must change because we operate on geometrically much larger entities. Also the way of application is different - candidates rejected at this stage can be later restored to the status of a figure. Instead of permanently removing, heuristics of this stage only mark figure candidates as rejected. This happens in the case of the candidates having incorrect aspect ratio, incorrect sizes or consisting only of horizontal lines (which is usually the case with mathematical formulas but also tables).

In addition to using the aforementioned heuristics, having clusters consisting of a mixture of textual and graphical operations allows applying new ones. During the next phase, we analyse the type of operations rather than their relative location. In some cases, steps described earlier might detect objects that should not be considered a figure, such as text surrounded by a frame. This situation can be recognised by the calculation of a ratio between the number of graphical and textual operations in the content stream of a figure candidate. In our approach we have defined a threshold which indicates which figure candidates should be rejected because they contain too few graphics. This allows removing for instance blocks of text decorated with graphics for aesthetic reasons. The ratio between numbers of graphical and textual operations is smaller in the case of tables than in the case of figures so extending the heuristic with an additional threshold could improve the table/figure distinction. Another heuristic analyses the ratio between the total area of graphical operations and the area of the entire figure candidate.

Subsequently, we mark as deleted figure candidates containing horizontal lines as the only graphical operations. These candidates describe tables or mathematical formulas which have survived previous steps of the algorithm. Tables can be reverted to the status of figure candidates in later stages of processing.

Figure candidates that survive all the phases of filtering are finally considered to be figures. Figure 3.4 shows a fragment of a publication page with indicated text areas and final figure candidates detected by the algorithm.

3.3.6 Detection and Matching of Captions

The input of the part of the algorithm responsible for detecting figure captions consists of previously determined figures and all text clusters. The observation of scientific publications shows that typically captions of figures start with a figure identifier (for instance see the grammar for figure captions proposed by Bathia et al. [8]).

The identifier usually starts with a word describing a figure type and is followed by a number or some other unique identifier. In the case of more complex documents, the figure number might have a hierarchical structure reflecting for example the chapter number. The set of possible figure types is very limited. In the case of HEP publications, the most usual combinations include words “Figure”, “Plot” and different variations of their spelling and abbreviating.

During the first step of the caption detection, all text clusters from the publication page are tested for the possibility of being a caption. This consists of matching the beginning of the text contained in a textual cluster with a regular expression determining what a figure caption is. The role of the regular expression is to elect strings starting with one of the predefined words, followed by an identifier or beginning of a sentence. The identifier is subsequently extracted and included in the metadata of a caption. The caption detection has to be designed to reject paragraphs of the type “Figure 1 presents results of (...)”. In order to achieve this, we reject the possibility of having any lower case text after the figure identifier.

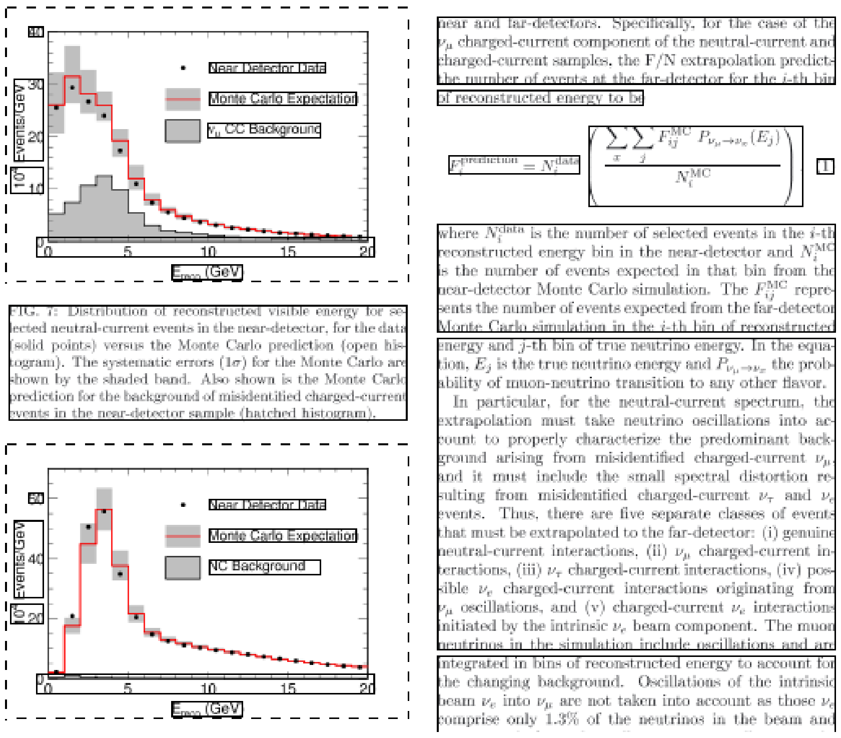


FIG. 7: Distribution of reconstructed visible energy for selected neutral-current events in the near-detector, for the data (solid points) versus the Monte Carlo prediction (open histogram). The systematic errors (1σ) for the Monte Carlo are shown by the shaded band. Also shown is the Monte Carlo prediction for the background of misidentified charged-current events in the near-detector sample (hatched histogram).

near and far-detectors. Specifically, for the case of the ν_μ charged-current component of the neutral-current and charged-current samples, the F/N extrapolation predicts the number of events at the far-detector for the i-th bin of reconstructed energy to be

$$F_i^{\text{prediction}} = N_i^{\text{data}} \left(\frac{\sum_x \sum_j F_{ij}^{\text{MC}} \, P_{\nu_\mu \to \nu_x}(E_j)}{N_i^{\text{MC}}} \right) \quad (1)$$

where N_i^{data} is the number of selected events in the i-th reconstructed energy bin in the near-detector and N_i^{MC} is the number of events expected in that bin from the near-detector Monte Carlo simulation. The F_{ij}^{MC} represents the number of events expected from the far-detector Monte Carlo simulation in the i-th bin of reconstructed energy and j-th bin of true neutrino energy. In the equation, E_j is the true neutrino energy and $P_{\nu_\mu \to \nu_x}$ the probability of muon-neutrino transition to any other flavor.

In particular, for the neutral-current spectrum, the extrapolation must take neutrino oscillations into account to properly characterize the predominant background arising from misidentified charged-current ν_μ, and it must include the small spectral distortion resulting from misidentified charged-current ν_τ and ν_e events. Thus, there are five separate classes of events that must be extrapolated to the far-detector: (i) genuine neutral-current interactions, (ii) ν_μ charged-current interactions, (iii) ν_τ charged-current interactions, (iv) possible ν_e charged-current interactions originating from ν_μ oscillations, and (v) charged-current ν_e interactions initiated by the intrinsic ν_e beam component. The muon neutrinos in the simulation include oscillations and are integrated in bins of reconstructed energy to account for the changing background. Oscillations of the intrinsic beam ν_e into ν_μ are not taken into account as those ν_e comprise only 1.3% of the neutrinos in the beam and

Figure 3.4: A fragment of the PDF page with boxes around every detected text area and figure candidate. Dashed rectangles indicate figure candidates. Solid rectangles indicate text areas.

Having the set of all the captions, we start searching for corresponding figures. All previous steps of the algorithm take into account the division of a page into text columns (see Section 3.3.2 about the layout detection). When matching captions with figure candidates, we do not take into account the page layout.

Matching between figure candidates and captions happens at every document page separately. We consider every detected caption once, starting with those located at the top of the page and moving downwards towards the end. For every caption we search figure candidates lying nearby. First we search above the caption and in the case of failure, we move below the caption. We take into account all figure candidates, including those rejected by heuristics.

In the case of finding multiple figure candidates corresponding to a caption, we merge them into a single figure, treating previous candidates as subfigures of a larger figure. We also include small portions of text and graphics previously rejected from figure candidates that lie between a figure and a caption and between different parts of a figure. These parts of text usually contain identifiers of the subfigures. The amount of unclustered content that can be included in a figure is a parameter of the extraction algorithm and is expressed as a percentage of the height of the document page.

It might happen that captions are located in a completely different location, but this case is rare and tends to appear in older publications. The distance from the figure is calculated based on the page geometry. The captions should not be too distant from the figure.

3.3.7 Generation of the Output

The Graphical Representation of Figures

The choice of the format in which data should be saved at the output of the extraction process should take into account further requirements.

The most obvious use case of displaying figures to end users in response to text-based search queries does not yield very sophisticated constraints. A simple raster graphics annotated with captions and possibly some extracted portions of metadata would be sufficient. Unfortunately, the process of generating raster representations of figures might lose many important pieces of information that could be used in the future for an automatic analysis.

In order to store as much data as possible, apart from storing the extracted figures in a raster format (e.g., Portable Network Graphics (PNG)), we also decided to preserve their original vector character. Vector graphics formats, similarly to PDF documents, contain information about graphical primitives. Primitives can be organised in larger logical entities. Sometimes rendering of different primitives leads to a modification of the same pixel of resulting image. Such a situation might happen for example when circles are used to draw data points lying nearby on the same plot. In order to avoid such issues, we convert figures into Scalable Vector Graphics (SVG) [23] format.

On the implementation level, the extraction of vector representation of a figure proceeds in a manner similar to regular rendering of a PDF document. The interpreter preserves the same elements of the state and allows their modification by transformation operations. A virtual canvas is created for every detected figure. The content stream of the document is processed and all the transformation operations are executed modifying the interpreter state. The textual and graphical operators are also interpreted, but they affect only the appropriate canvas of the figure to which the operation belongs. If a particular operation does not belong to any figure, no canvas is affected. The behaviour of graphical canvases used during the SVG generation is different from the case of raster rendering. Instead of creating graphical output, every operation is transformed into a corresponding primitive and saved within a SVG file.

The PDF format was designed in such a manner that the number of external dependencies of a file is minimised. This design decision led to the inclusion of the majority of fonts in the document itself. It would be possible to embed font glyphs in the SVG file and use them in order to render strings. However, for the sake of simplicity, we decided to omit font definitions in the SVG output.

A text representation is extracted from every text operation and the operation is replaced by a SVG text primitive with a standard font value. This simplification affects how the output looks like, but the amount of formatting information that is lost is minimal. Moreover, this does not pose a problem as vector representations are intended to be used during automatic analysis of figures rather than for displaying purposes. A possible extension of the presented method could involve embedding complete information about used glyphs.

The External Metadata

This section describes how the metadata generated in the course of PDF extraction can be used to generate RDF triples using the HFO vocabulary. The metadata created by the PDF extractor is external to the figure (i.e. it describes a figure in the context of the publication from which it has been extracted). It also describes a figure as a single entity rather than annotating particular parts. Table 3.1 lists the produced metadata fields. It also shows the corresponding parts of the ontology. We omit the details of the material description, which are similar for all types of objects.

The PDF extractor provides a very limited description of the type of a figure. Such a situation is a consequence of the fact that the produced metadata is external to a figure. Initially, a figure is annotated with only a general type: Figure or Table. Later analysis based on the content of the figure (described in the following chapter) allows the assignment of a more specific type.

Table 3.1: Automatically generated metadata and matching properties in the ontology

Extracted metadata	**Related concept in the ontology**
Caption of a table/figure	The Dublin Core title attribute
Location of a caption inside a publication	Part of the material description
Type (table or figure)	The class inside HFO
Identifier within the original publication	hfo:hasName
Location within the original publication	Part of the material description
The complete text of a figure	hfo:hasFullText
References from the text of a publication	hfo:hasTextReference

3.4 Tuning and Testing

The extraction algorithm described here has been implemented in Java and tested on a random set of scientific articles coming from the INSPIRE repository. The testing procedure has been used to evaluate the quality of the method, but also allowed tweaking the parameters of the algorithm in order to maximise outcomes.

3.4.1 Preparation of the Testing Set

In order to prepare the testing set, we randomly selected 207 documents stored in INSPIRE. In total, these documents consisted of 3728 pages which contained 1697 figures altogether.

The records have been selected according to a uniform probability distribution across the entire record space. This way, we have created a collection that is representative for the entire INSPIRE including historical entries.

Currently, INSPIRE consists of: 1,140 records describing publications written before 1950; 4,695 between 1950 and 1960; 32,379 between 1960 and 1970; 108,525 between 1970 and 1980; 167,240 between 1980 and 1990; 251,133 between 1990 and 2000; and 333,864 in the first decade of XXIst century. In total, up to July 2012, INSPIRE manages 952,026 records. It can be seen that the rate of growth has increased with time and most of INSPIRE documents come from the last decade.

The results on such a testing set should accurately estimate the efficiency of extraction for existing documents but not necessarily for new documents, being ingested into INSPIRE. This is because INSPIRE contains entries describing old articles which were created using obsolete technologies or scanned and encoded in PDF. The extraction algorithm is optimised for born-digital objects. In order to test the hypothesis that the extractors provides better results for more recent papers, the testing set has been split into several subsets. The first set consists of publications published before 1980. The rest of the testing set has been split into subsets corresponding to decades of publication.

In order to simplify the counting of correct figure detections and to provide a more reliable execution and measurement environment, every testing document has been split into a number

	–1980	1980–1990	1990–2000	2000–10	2010 – 12
number of existent figures	114	60	170	783	570
number of correctly detected figures	59	53	164	703	489
number of incorrectly detected figures	26	78	65	40	73
total number of pages	85	136	760	1919	828
number of correctly processed pages	20	44	712	1816	743

Table 3.2: Results of the test execution

of PDF documents consisting of a single page. Subsequently, every single page document has been manually annotated with the number of figures appearing inside.

3.4.2 Execution of the Tests

The efficient execution of the testing was possible thanks to a special script executing the plots extractor on every single page separately and then computing the total number of successes and failures. The script allows the execution of tests in a distributed heterogeneous environment and allows dynamic connection and disconnection of computing nodes. In the case of a software failure, the extraction request is resubmitted to a different computation node, allowing the avoidance of problems related to a worker node configuration rather than to the algorithm implementation itself. The design of the script has been inspired by the architecture of Apache Hadoop [41].

Using aggregated numbers from all extracted pages allowed us to calculate efficiency measures of the extraction algorithm. As quality measures, we used recall and precision [5]. Equation 3.4.1 shows the definitions of both of them. In every place where we needed a single comparable quality measure rather than two semi-independent numbers, we have used a harmonic average of the precision and the recall[5].

$$\text{recall} = \frac{\#\text{correctly extracted figures}}{\#\text{figures present in the test set}} \tag{3.4.1}$$

$$\text{precision} = \frac{\#\text{correctly extracted figures}}{\#\text{extracted figures}} \tag{3.4.2}$$

$$\text{harmonic average} = \frac{2}{\frac{1}{\text{precision}} + \frac{1}{\text{recall}}} \tag{3.4.3}$$

Table 3.2 summarises the results obtained during the test execution for every subset of our testing set. Figure 3.5 shows the dependency of recall and precision on the time of publication. The extractor parameters used in this test execution were chosen based on intuition and small number of manually triggered trials. In the next section we describe an automatic tuning procedure we have used to find the most optimal algorithm arguments.

It can be seen that, as expected, the efficiency increases with the increasing time of publication. A total recall and precision for all samples since 1990, which constitutes a majority of the INSPIRE corpus, were both 88%.

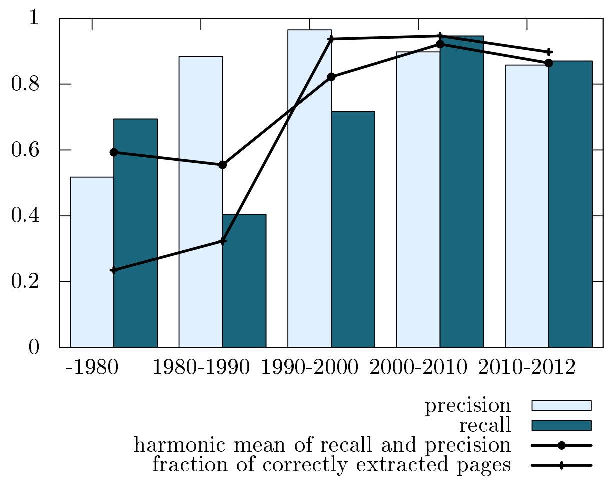


Figure 3.5: Recall and Precision as functions of decade of the date of the publication.

Precision and recall based on the correctly detected figures do not give a full image of the algorithm efficiency because the extraction has been executed on a number of pages not containing any figures. The correctly extracted pages not having any figures do not appear in the recall and precision statistics because in their case the expected and detected number of figures are both equal to 0.

Besides recall and precision, Figure 3.5 depicts also the fraction of pages which have been extracted correctly. Taking into account the samples since 1990, 3,271 pages out of 3,507 have been detected completely correctly, which makes 93% success rate counted by number of pages. As it can be seen, this measure is higher than both the precision and the recall.

The analysis of the extractor results in the case of failure shows that in many cases, even if results are not completely correct, they are not far from the expectation. There are different reasons of the algorithm failing. Some of them may result from non-optimal choice of algorithm parameters, others from document layout being too far from the assumed one. In some rare cases, even manual inspection of the document does not allow an obvious identification of figures.

3.4.3 The Automatic Tuning of Parameters

In the previous section we have shown results obtained by executing the extraction algorithm on a sample set. During this execution we were using extractor arguments which seemed to be the most correct based on our observation but also on other research [12] (typical sizes of figures, margin sizes etc.). This way of algorithm configuration was useful during the development, but is not likely to yield the best possible results. In order to find better parameters, we have implemented a method of automatic tuning. The metrics described in the previous section (recall and precision) provided us with a good method of measuring the efficiency of the algorithm running based on given parameters.

The choice of optimal parameters can be relative to the choice of documents on which the extraction is to be performed. The way in which the testing set has been selected, allowed us to use it as representative for the HEP publications. In order to tune the algorithm, we have used a described subset of testing set from the previous step as a reference. The subset consisted of all entries created after 1990. This allowed us to minimise the presence of scanned documents which, by design, cannot be correctly processed by our method.

The adjustment of parameters has been performed by a dedicated script which has executed the extraction using various parameter values and has read results. The script has been configured with a list of tunable parameters together with their type and allowed values range. Additionally, the script had knowledge of the believed best value, which was the one used in previous testing.

In order to decrease the complexity of training, we have made several assumptions about the parameters. These assumptions are only an approximation of real nature of parameters but the practice has shown that they are good enough to permit the optimisation:

- We assume that the precision and recall are continuous with respect to the parameters. This allows us to assume that efficiency of the algorithm for parameter values close to a given one will be close. The Optimisation has proceeded by sampling the parametric space in a number of points and executing tests using the selected points as parameter values. Having N parameters to optimise and dividing the space of every parameter into M regions leads to the execution of M^N tests. Execution of every test is a timely operation due to the size of the training set.

- We assume that parameters are independent from each other. This means that we can divide the problem of finding an optimal solution in the N-dimensional space of N configuration arguments into finding N solutions in 1-dimensional subspaces. Such an assumption seems to be intuitive and considerably reduces the number of necessary tests from $O(M^N)$ to $O(M \cdot N)$, where M is the number of samples taken from a single dimension.

In our tests, the parametric space has been divided into 10 equal intervals in every direction. In addition to checking the extraction quality in those points, we have executed one test for the so-far best argument. In order to increase the level of fine-tuning of the algorithm, each test has been re-executed in the region, where chances of finding a good solution were considered the highest. This consisted of a region centred around the highest result and having a radius of 10% of the parameter space.

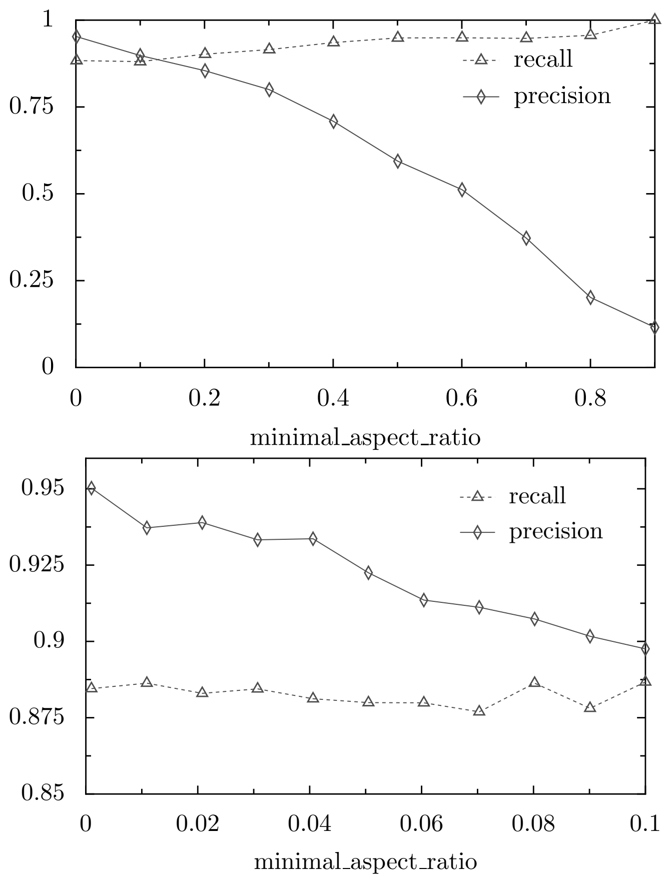


Figure 3.6: Effect of the minimal aspect ratio on precision and recall

Figure 3.6 and Figure 3.7 show the dependency of the recall and the precision on an algorithm parameter. The parameter depicted in Figure 3.6 indicates what minimal aspect ratio the figure candidate must have in order to be considered a correct figure. It can be seen that tuning this heuristic increases the efficiency of the extraction. Moreover, the dependency of recall and precision on the parameter is monotonic which is the most compatible with the chosen optimisation method.

The parameter of Figure 3.7 specifies which fraction of the area of the entire figure candidate has to be occupied by graphical operations. This parameter has a lower influence on the extraction efficiency. Such a situation can happen when more than one heuristic influences the same aspect of the algorithm. This is contradictory with the assumption of parameter independence, but we have decided to use the present model for the simplicity.

After executing the optimisation algorithm, we have managed to achieve the recall of 94.11% and the precision of 96.6% which is a considerable improvement with respect to previous results

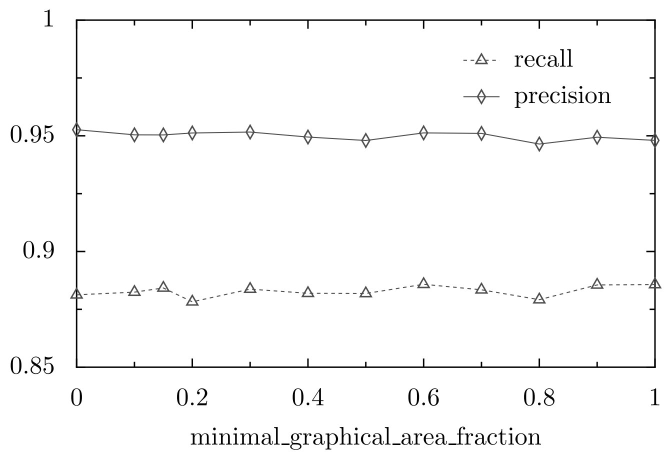


Figure 3.7: Effect on the precision and recall of the area fraction occupied by graphical operations

of 88%.

3.5 Summary

This chapter has presented a method of extracting figures from scientific publications in a machine readable format, which is the main step towards the development of services enabling access and search for images stored in scientific digital libraries. Our method of detecting fragments of PDF documents that correspond to figures is based on a series of observations on the character of publications. The code of the implemented prototype is freely available online[5].

However, additional work is needed on heuristics allowing the correct detection of figures, what was showed by tests. Also the performance should be reevaluated after we have a large set of correctly annotated figures, confirmed by users of our system.

The heuristics used by the algorithm are based on a number of numeric parameters which we have tried to optimise using automatic techniques. The tuning procedure has made several arbitrary assumptions on the nature of the dependency between parameters and extraction results. A future approach to the parameter optimisation, requiring much more processing, would involve the execution of a genetic algorithm [47] which would treat the parameters as gene samples. This could potentially allow a discovery of a better parameter set because fewer assumptions would be imposed on the parameters. A vector of algorithm parameters could play the role of a gene and random mutations could be introduced to previously considered and subsequently crossed genes. The evaluation and selection of surviving genes could be performed by the usage of the metrics described in section 3.4.2.

Another approach for improving the quality of the parameters tuning could involve extending the present algorithm by a discovery of mutually dependent parameters and use special techniques (relaxing the made assumptions) to fine-tune in subspaces spanned by them.

All of our experiments have been performed using a corpus of publications from HEP. If we decided to use the extraction algorithm on a different corpus of documents, the algorithm should first be tuned on a representative sample set taken from this corpus. For the area of HEP, we can also consider preparing several sets of execution parameters varying by decade of document publication or by other easy to determine characteristics. Subsequently, we could decide which extraction method to run, based on those metrics.

In addition to a better tuning of existing heuristics, there are improvements which can be made at the level of the algorithm. As an example of such improvements we could mention extending the process of clustering text parts. In the current implementation, the margins by which textual operations are extended during the clustering process are fixed as algorithm parameters. This approach proved to be robust in most cases. In fact, distances between text lines tend to be different depending on the currently utilised style. Every text portion tends to have one style that is dominating. An improved version of the text-clustering algorithm

[5]`https://github.com/ppiotr/PDFPlotsExtractor`

could use local rather than global properties of the content. Not only this would allow correctly handling the entire document written using different text styles, but also it would help to manage cases of single paragraphs differing from the rest of the content.

Another important, not implemented yet, improvement related to figure metadata is the automatic extraction of figure references from the text content. Important information about figure content might be stored in the surroundings of the place where publication text refers to a figure. Furthermore, the meta-data could be extended by the usage of some type of classifier which would assign a graphics type to the extracted result. Currently, we are only distinguishing between tables and figures based on simple heuristics involving number and type of graphical areas and the text inside the detected caption. In the future, we could detect line-plots from photos, histograms and so on. Such a classifier could be implemented using Artificial Intelligence techniques such as Support Vector Machines (SVM) [50].

Finally, partial results of the figures extraction algorithm might be useful in performing other PDF analyses:

- The usage of clustered text areas could allow a better interpretation and indexing of textual content stored in digital libraries with full text access. Clusters of text tend to describe logical parts like paragraphs, section and chapter titles and so on. A simple extension of the current schema could allow the extraction of predominant formatting style of the text encoded in a page area. Text parts written in different styles could be indexed in a different manner giving for instance more importance to segments written with larger font.
- In section 3.3.5 we mentioned that the algorithm detects not only figures, but also tables. A heuristic is being used in order to distinguish tables from different types of figures. Our present effort concentrates on correct treatment of figures, but a useful extension could allow extraction of different types of entities. For instance, another common type of graphics ubiquitous in HEP documents are mathematical formulas. Thus, in addition to figures, it would be important to extract tables and formulas in structured format allowing a further processing.

Chapter 4

Extraction of Information from Figures

4.1 Introduction

The previous chapter described a method of extracting figures from scientific publications. The extraction process generates elements of description of figures. However, all the metadata generated in the course of automatic figure-extraction describes these figures in the context of a publication. Even if the extraction algorithm has to process the elements which form a figure, the internal logical structure and meaning of these are at this stage completely ignored. The construction of an Information Retrieval (IR) system operating on figures requires knowledge about the content of these. The elements of the description of figures as parts of the publication text carry indications about what type of data is encoded inside a graph. However, this information is usually of a very general and limited nature. Processing a complete description of figures has the potential of increasing the accuracy of such IR systems.

The remainder of this chapter is structured as follows: Section 4.2 describes the existing systems managing data encoded in plots and depicts existing approaches to the retrieval of information from images. Section 4.3 presents the overview of the architecture of a system prototype. In the following Section 4.4 we describe a method which has been used to implement a module of the semantics-extraction system and to demonstrate the feasibility of the system. Finally, Section 4.6 describes the direction of future developments for creating a complete semantics-extraction system.

4.2 Existing Approaches for Analysing the Content of Images

There exist different approaches which can be used to obtain the content of figures. Different approaches are optimal in different scenarios, depending on the amount of available resources and the character of the data. This section presents some of the existing approaches for gathering data behind figures.

HepData[1] [2] is a project which gathers tables of data coming from publications. These data appears either in the text of publications or in figures and tables. The body of HepData is either provided by authors, or manually extracted from selected publications. Publication and dataset entries are extended with a supplementary comment and with the location inside the original publication. The location indicates the type of artifact which is described by the data (it allows recognising which datasets are extracted from plots). At the moment of writing, HepData contains data from about 7000 selected publications and is widely used by the HEP community. HepData is a potential source of data which could be used to obtain data behind HEP figures managed in INSPIRE. A more complete description of HepData can be found in Chapter 6 containing a description of the integration of HepData descriptions as INSPIRE bibliographical records. Also in other domains, there exist initiatives of gathering and publishing the data behind figures. Journals of American Astrophysical Society accept [3] tabular data, which can be published along with the figures. A common characteristic of all such repositories of data is that the quality of the data is high (as it is manually reviewed), however the datasets cover a small fraction of the entire domain. INSPIRE, which is a domain digital library of HEP, contains over 1.230.000 distinct publications. This number is much higher than the number of publications described by the HepData project. The long-term objective of INSPIRE is to provide description and search over the complete set of figures stored in all HEP publications. Relying only on the data gathered by HepData would not allow achieving this goal. According to our best knowledge, there is not a service giving access to a complete set of figures together with the description of their content. Besides the above difficulty, the data model of HepData is very rigid, concentrating on the description of Monte Carlo simulations and having difficulties when describing more complex plots, which might appear in HEP publications.

Another possibility of gathering descriptions of figures involves asking a large group of participants to provide the data. Tools like Mechanical Turk [32] allow engaging a high number of users to perform a simple task. In the case of mining descriptions of figures, users could be presented a task of describing (using a controlled vocabulary), what is located in a figure.

[1]http://durpdg.dur.ac.uk/
[2]`http://feynml.hepforge.org`
[3]http://aas.org/posts/blog/2012/09/data-behind-figures

However, the task of describing scientific graphs is a rather specialised task requiring some experience with scholarly publications. It cannot be assumed that the users of crowdsourcing tools would be able to perform the task correctly. Moreover, parts of plot annotation are a highly mechanical task consisting of inputting many similar numbers. Machines are much more robust when managing this type of situations.

A common characteristic of all the aforementioned approaches, which require the human intervention, is that the process requires constant attention from the side of operators and as such is not self-sustainable. Those scenarios usually require a relatively low amount of effort in order to establish the data-gathering process. However, maintaining the processing of all newly-acquired documents requires constant human work. An algorithmic approach requires a much higher effort at the phase of establishing the process, however if it is designed correctly, can subsequently run without much of the intervention. In such a case, manual interventions could be saved only for the cases in which the automatic process failed.

Extracting meaning of figures is closely related to the well-studied problem of Artificial Intelligence (AI) - the detection of objects depicted inside a given graphics. Description of the meaning of images has long been an object of interest in Computer Vision (CV). However, because of the immense complexity of the problem, a method for describing an arbitrary image in a formal manner does not exist. Instead, there are solutions to smaller problems.

One of the main problems of Computer Vision concerns the detection of a known object inside an image. Wavelet-decomposition has been studied [36, 51, 16] as a tool for the resolution-independent detection of a known pattern inside raster images. There are also techniques using neural networks [46]. The above techniques are proven to be efficient when detecting a general class of objects, like “a face”, “a car” and so on. In all the mentioned cases, the algorithm is initially trained by providing a set of examples before being capable of recognising new appearances.

The previously-described techniques operate on raster images. However, there exist methods which first preprocess an image in order to extract its features. Such features can consist of the location of contours which are present in a picture. The previously-extracted features can form a basis for the extraction of entities. The Hough Transform [18] is a technique which allows detecting analytic curves inside pictures which already have their features extracted. The shapes are detected in a rotation-invariant manner, which made the technique useful for the vectorisation of raster images. Later, the technique has been generalised to allow the detection of more complex shapes [6]. The nature of this technique makes it applicable in the cases where images contain very distinct boundaries (like for example in the detection of hand-writing or line-drawings). In the case of images with a lot of gradual tone-progression, the technique cannot be applied directly. The Generalised Hough Transform is a good candidate for the usage with vector graphics, where the initial feature-extraction (edge detection) can be replaced by conversion of existing shapes.

Object-detection typically serves as the initial step in more complex image-recognition algorithms (Especially in the case of raster graphics with reach meaning). Techniques able to detect appearances of certain classes of objects are usually incapable of recognising a particular instance of such an object. Detection of instances is also a well-studied problem of the CV. Solutions to this problem allow, for example, to detect who is the person whose face has been previously detected or what is the model of a car whose presence has been discovered by a wavelet-based technique. A popular approach to this category of problems involves usage of statistical techniques and the interpretation of an image as a vector in certain n-dimensional space. In the case of face-recognition, the image can be for example first cropped to contain only the face, subsequently converted to grey-scale image in a standardised resolution. After such a conversion, every image consists of the same number of pixels. Each of these pixels is in turn represented by a single number indicating the intensity of grey. If each image is represented by a raster of width W and height H, it can be represented also as a vector in a $W \times H$ dimensional real space. As such, it can be processed using methods like Primary Component Analysis (PCA) [38], or Singular Value Decomposition (SVD), which allow to select the most important features and to reduce the dimensionality of used representation. This in turn makes the usage of standard clustering and matching techniques of Artificial Intelligence possible by reducing the volume of the data and by removing irrelevant features.

The results of all previously-mentioned methods consist of isolated sets of descriptions indicating that a certain type of object or a certain individual are depicted at a certain position of an image. However, in the case of images, typically the sum of meanings of separate elements do not give the total meaning of the image. Different constituent parts of an image are related to each other and the nature of those relations is meaningful. Semantic techniques, like the Description Logic (DL) [4] and in a more implementation-specific context OWL and RDF ontologies, allow describing such relations between different classes of object and different individuals. Such methods, combined with the aforementioned image-recognition algorithms, can be used to unveil the complete meaning of an image. Dasiopoulou [15] presents for example a technique for using fuzzy DL combined with statistical image detection in order to reason about the content of a scene.

At first, the image is segmented into a number of areas, which are classified using standard statistical techniques into a number of categories (“water”, “grass”, “person” . . .). The difference with typical usage of such classifiers lies in the fact that the algorithm takes into account how certain a classifier was when detecting a certain element. Typically, statistical classifiers can provide such values. A semantic knowledge base is used to reason about the content of the entire image based on the previous descriptions. The knowledge base (Terminological Box (TBox) in the terminology of Description Logic) contains rules that describe which types of larger concepts are likely to be present in an image if smaller concepts appear. For example “countryside buildings scene is a subset of scenes which contain a building and contain grass”.

Additionally, the semantics can describe spatial relations between different objects. The semantic reasoning system finds evaluations of variables corresponding to different segments of the image into concepts. Each of these evaluations can be assigned a probability with which it is true, based on the input probabilities provided by the statistical classifiers. Every final evaluation should be consistent with the rules present in the TBox. In the case of scientific plots, spatial relation between different elements is usually the same, making similar techniques of smaller applicability. However, diagrams tend to represent relations between objects. A content-extraction system for graphs, could take advantage of described in [15] formalised rules.

A thorough detection of the meaning of an arbitrary picture is a difficult challenge, which at the moment of writing is far from being solved. It is closely related to the problem of making arbitrary information readable and processable by machines, which is one of the big goals of the Artificial Intelligence. The creation of a domain-specific system which would be capable of extracting and processing information of a particular domain is a much smaller problem. This work tries to advance the border between what can be understood by machines and what cannot be by attempting to extract the meaning of HEP figures.

There exits attempts to extract knowledge from graphs [35, 9, 31]. However, the description is very general and limited to the most important features. Moreover, the extraction using raster images does not exploit all the potential of the data stored in HEP publications. The automatic procedure for extracting figures from HEP publications (described in Chapter 3) provides vector graphics representations of the figures. Processing the vector graphics automatically solves the problems encountered by other projects which aim at the extraction of meaning from graphs [35] and at the same time, it decreases the error generated by the automatic extraction process. In existing methods, a lot of efforts have been made to identify and disambiguate intersecting entities. When processing vector images, such a disambiguation is unnecessary as all the entities are described separately.

4.3 Architecture of the Feature Extraction System

The techniques described in the previous section tend to facilitate the discovery of a particular type of content inside an image. This section describes the extraction of figure's meaning from a more architectural perspective, dealing with the complexities of figures appearing in the domain of HEP. HEP publications can contain various types of figures, examples of them can be found in the Introduction chapter of this work. Every type of a figure encodes different information and has a different internal structure. Figure 4.1 shows a very general overview of the architecture of a system capable of extracting semantics from arbitrary graphics. The modular design allows the independent implementation of parts that are responsible for processing different types of content, decreasing the inter-dependence between different extractors.

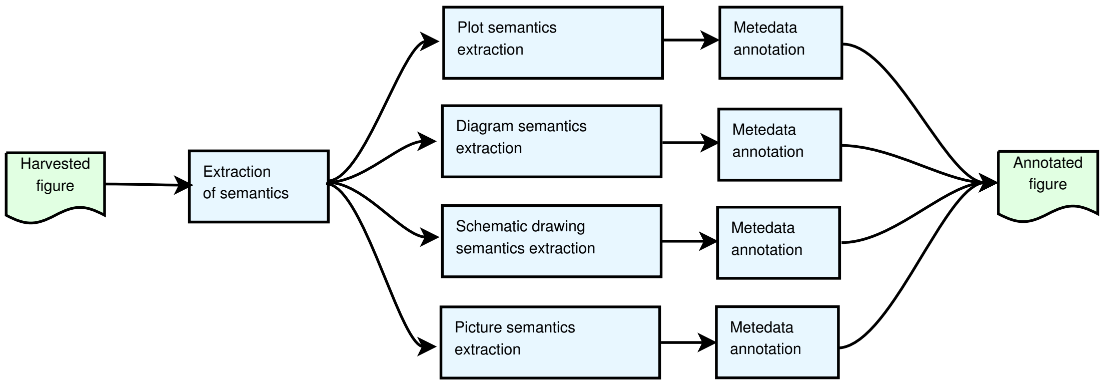


Figure 4.1: Overview of the process of extraction of semantics of an arbitrary figure

Instead of trying to detect the type of a figure and subsequently executing an appropriate extraction module, the semantics extractor passes the input figure to every single module. Modules should be implemented in such a way that output is generated only if the figure contains relevant content, leaving the output empty otherwise. This content does not have to cover the entire canvas of a figure. Such a design increases the flexibility of the extraction procedure. In many cases, figures can contain elements typical for different types of content. For instance, a plot can be combined with a diagram inside the same graphics. In such a case, unambiguous identification of a figure type would be impossible (preventing the figure from being passed to a single best extractor). On the other hand, executing separate extraction modules would allow the separate description of the subfigure depicting a diagram and the one depicting a plot.

Such a parallel design is possible because the extraction of different figure types is independent. However, it cannot be applied for the extraction of different features of a single figure type. Throughout the remainder of this chapter, we will concentrate on a single example of an extraction module. In order to prove the feasibility of the data extraction from figures, we have implemented a prototype of a module extracting semantics of plots.

4.4 Extraction of Features from Plots

Plots are the most frequent among different types of figures present in HEP publications and as such, they are of special interest for a HEP-specific IR system. Before describing the process of plot-semantics extraction, we analyse the structure of a typical plot and present the internal architecture of the extractor module.

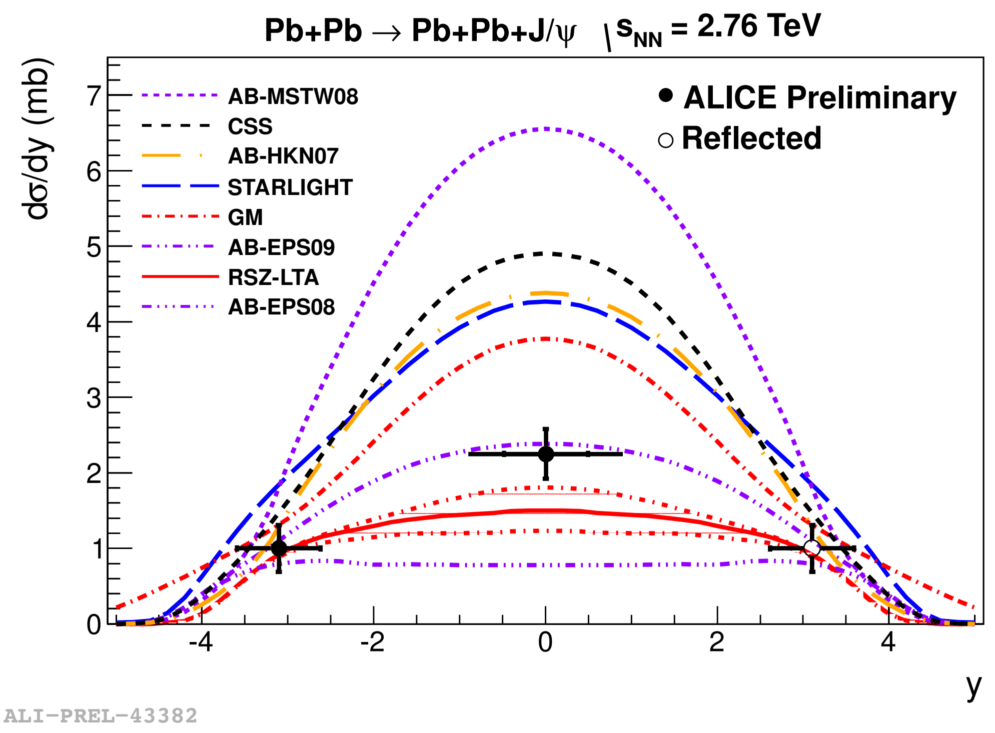


Figure 4.2: A typical plot being subject to the extraction procedure. This example comes from http://inspirehep.net/record/1217556

4.4.1 The Anatomy of a Plot

Figure 4.2 depicts a typical 2-dimensional plot appearing in a HEP publication. Most of the plots share common features, which a module extracting the semantics of them should be able to extract and describe. Chapter 2 delves into the details of the description of different parts of a plot. At this place, we only identify these elements in order to justify the design of the extraction system.

The most prominent feature of a plot is the corresponding coordinates system, which forms a basis for the interpretation of the encoded data. Coordinate system contains a number of axes (in the case of a 2-dimensional plot, typically the number of axes is equal to 2). Axes are associated with value-indicating ticks and labels. Besides coordinate systems, plots containing data lines and legends. Some plots contain a supplementary textual description of their content. The aforementioned features of a plot are closely related to each other. The results of extracting certain parts influence the process of extracting others. Figure 4.3 shows the internal structure of a module capable of extracting the semantics of figures. Every module has access to the complete representation of the original figure together with a subset, which has not yet been

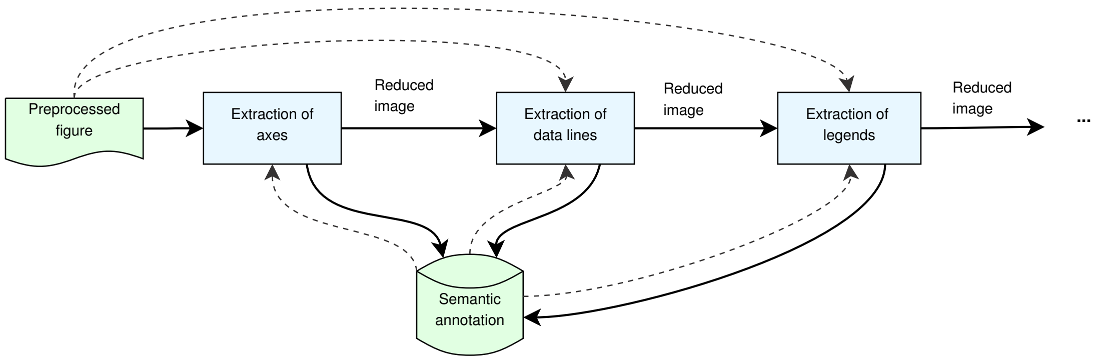


Figure 4.3: A general architecture of an extraction module for plots

detected by other modules. After detecting an entity (like for example an axis), the module can remove the graphical primitives representing such a feature visually. This allows decreasing the ambiguity during the further processing and increases the consistence of the final description by not allowing the same graphical objects to describe different logical parts. In theory, the complete extraction process detecting all the present features of a plot should produce an empty reduced representation.

Besides the figure, every module has also access to the annotation of already-detected parts. The possibility of reading the existing metadata is a source of dependencies between modules. At the beginning of the extraction, the annotation knowledge base is empty. After finishing the processing, the knowledge base should be populated with a complete description of a figure. The solid lines of Figure 4.3 depict the typical information flow in the extraction module. The dashed lines present the additional possible flow of data.

The extraction process from Figure 4.3 is depicted in a linear manner. However, a concrete implementation could be parallelised. The dependencies between different parts of the extraction process form a graph. The detection of axis labels is for example strictly dependent on the results of the extraction of coordinate systems and as such, has to be executed afterwards. At the same time, the extraction of data lines is not dependent on the extraction of these labels. Figure 4.3 depicts an example of the extraction order. This order is not unique. Any order resulting from sorting the graph of dependencies in a topological manner [13] is correct. Moreover, the graph can be used to enable the parallel execution of different parts of the algorithm. However, in the environment where the extraction is performed on a large number of figures, the concurrent execution of complete linear figure-extraction tasks seems to have a larger potential of increasing performance. That is because the execution of complete figure-extraction tasks lacks the necessity of a too frequent synchronisation between different results.

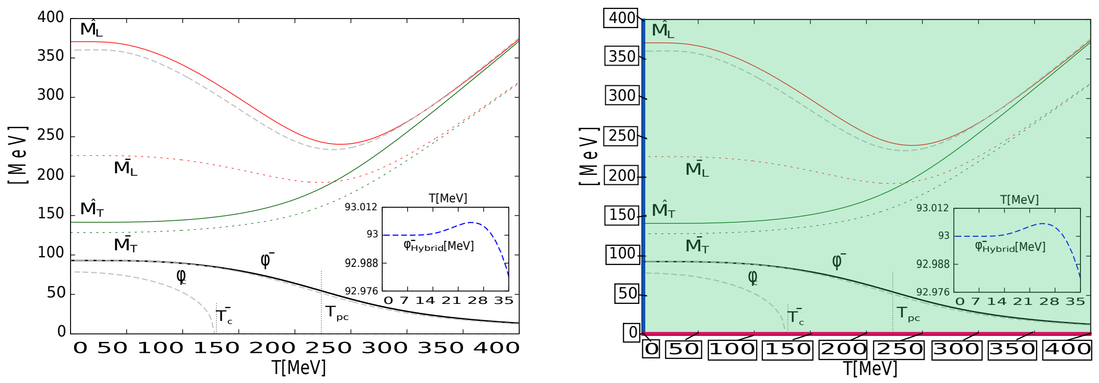


Figure 4.4: Graphical illustration of a sample coordinate system extracted from a plot.

4.4.2 Extraction of Coordinate Systems

This section describes the method which has been used to implement the extraction of coordinate systems. The extraction method detects on only the axes but also elements of the coordinate system annotation: axis ticks and labels. Additionally, detecting the coordinate systems allows recognising that a figure represents a plot. Figure 4.4 depicts a sample plot (on the left), together with a graphically annotated coordinate system. The coordinate system depicted in Figure 4.4 is not the only one present in the plot. There is a smaller coordinates system appearing inside the main plot.

Figure 4.5 depicts the general structure of the method of extracting coordinate systems. At the very beginning, we read the input document in Scalable Vector Graphics format. We interpret all the graphical primitives and use them to build our internal representation of a graph by discarding all curves and promoting all line segments which can possibly be part of a more complex shape to be separate line objects. We describe all the obtained line segments using a uniform coordinate system of the canvas on which the plot could be rendered.

Subsequently, we identify the candidates for coordinate system axes. We first identify pairs of lines which are approximately perpendicular related to each other and which intersect if extended by a small length unit. These pairs are considered candidates of a potential coordinates system. This method is similar to the technique used by Lu et Al. [35]. However, our approach is more general as it is not limited to the lines which are parallel to the borders of the canvas. We achieve this by the means of the quantisation of the parameter space, similar to the one used to implement the Hough transform [18]. This more general approach will be useful in the future, when applying our method to publications whose digital versions were created in the process of scanning. In such cases, the page can be rotated with respect to the document orientation, making coordinate systems inclined in an arbitrary angle.

Every coordinate system candidate is subject to further extraction of additional properties.

However, the number of previously selected candidates tends to be very high. The detailed processing of such a large set would be a major problem from the point of view of the efficiency of the extraction method. In order to diminish the number of potential candidates, we reject those which are very unlikely to form a coordinate system. This is achieved using a set of basic heuristics based on the length of the axis lines in relation to each other and to the dimensions of the canvas.

The candidates that survive the filtering phase are subject to the extraction of axis ticks. Usually, coordinate system axes are intersected by a number of short line intervals serving as a reference for the data described in the graph. The extraction algorithm detects such lines. However, ticks are not the only objects which intersect axes. Usually, also the data lines intersect with coordinate system axes, which might lead to a confusion. In order to address this difficulty, the tick detection algorithm selects intersecting lines which have uniform length (Two most numerous line lengths are considered) and tries to search for a regularity in the distance from the origin of the coordinate system understood as the point of intersection of the axes.

As the next step, the algorithm searches for bits of text laying sufficiently close to the detected ticks and recognises them as the labels. If after this step the number of labels is close to the number of axis ticks represented by longer lines, labels are reassigned to match only with them. This step is inspired by the observation that usually only part of axis ticks are labelled. If this is the case, longer ticks are typically matched with labels.

The coordinate system candidates annotated with axis ticks and with their labels are ready for the final classification. In order to decide which candidates describe axes, we decided to use a Support Vector Machines (SVM) [14] method. We have trained a statistical classifier using a sample set of extracted coordinate system candidates, classifying them as coordinate system and non-coordinate system objects. In order to use the statistical techniques, we had to express the considered candidates in a form of vectors from some n-dimensional real space. The dimensions which we considered suitable for the classification included:

1. The total number of detected axis ticks
2. The fraction of ticks which have labels assigned
3. The aspect ratio between both axes
4. The length of axis expressed as a fraction of the canvas dimensions
5. The ratio of arbitrary graphical primitives lying in the boundary of the coordinate system divided by the number of primitives lying in the boundary extended by a margin.

A more detailed description of the training procedure for the SVM classifiers can be found in Appendix C.

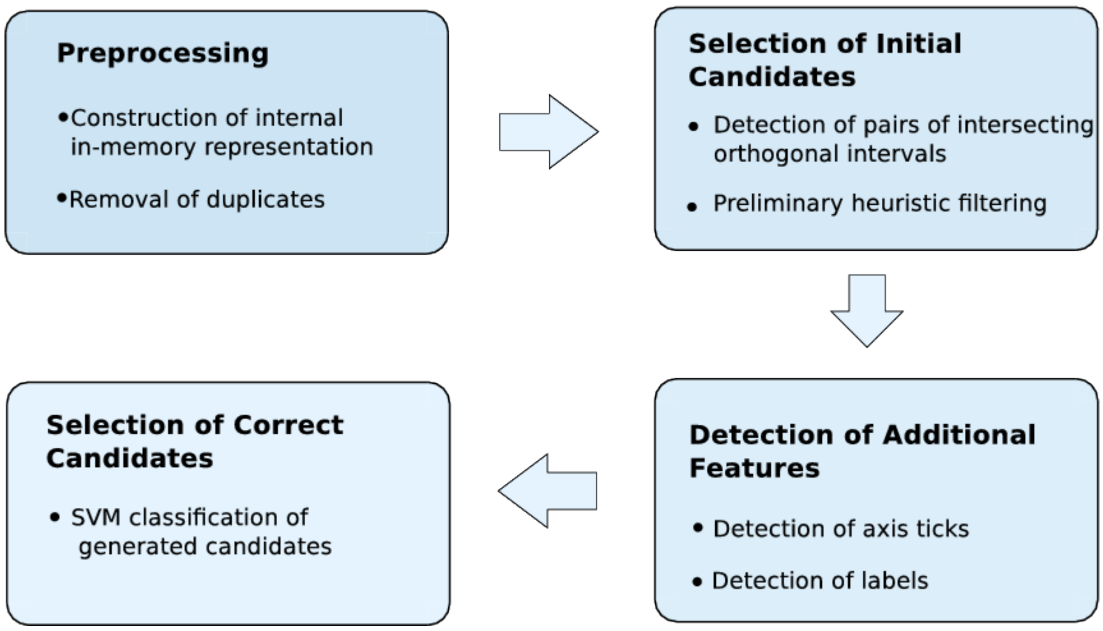

Preprocessing
•Construction of internal in-memory representation
•Removal of duplicates
Selection of Initial Candidates
• Detection of pairs of intersecting orthogonal intervals
• Preliminary heuristic filtering
Detection of Additional Features
• Detection of axis ticks
• Detection of labels
Selection of Correct Candidates
• SVM classification of generated candidates

Figure 4.5: The structure of the axis extraction.

The detection of coordinate systems does not guarantee that we are dealing with 2-dimensional plots. Sometimes the method yields parts of 3D coordinate systems with a single axis missing. However, the purpose of this prototype was to demonstrate the possibility of automatic extraction of semantics from plots. Further work should concentrate on expanding the implementation to different special cases, so that the majority of different plots of HEP would be covered.

4.4.3 Annotation with the HFO Ontology

This section summarises the metadata elements produced during the automatic extraction of axes. We relate these data to the elements of the ontology, which can be used to encode them.

The coordinate system extractor is capable of detecting the following entities in our ontology: CoordinateSystem, Axis, AxisLabel, MinorTick, MajorTick and TickLabel. The detected entities are added to the knowledge base together with the hfo:contains relations connecting them. Table 4.1 summarises the most important properties generated by the extractor and describes the concepts of the HFO ontology used to describe them. Similarly to the case of external properties, both general and material description elements are produced. However, at this point we describe only the concepts of the central ontology.

As mentioned earlier, certain bits of information can be derived from the output of extractors in a more indirect way. For example, if the output of the extractor yields the existence of

Table 4.1: The annotation elements generated by the coordinate system extractor

Extracted Metadata	**Related Concept in the Ontology**
Location and number of coordinate systems	CoordinateSystem entities, parts of material description (location of the origin of a coordinate system)
Location of axes	The material description
Location and types of axis ticks	MinorTick, MajorTick, hfo:atPosition, material description of the tick lines.
Location of tick labels	The material description
Text of tick labels	hfo:hasLabel, AxisTickLabel entity with the content
Location of axis labels	The material description

Table 4.2: Examples of labels associated with concepts from the HEP taxonomy

Concept of the HEP taxonomy	**Label**
HEPOnt:trackdataanalysis	Track data analysis
HEPOnt:Composite.strangenessmagnetic-momentcalculated	strangeness: magneticmoment: calculated
HEPOnt:Composite.Antixi-electroproduction	Antixi-: energy
HEPOnt:Composite.Altarelli-Parisiequationsolution	Altarelli-Parisi equation: solution
HEPOnt:Composite.antineutrino/ taucascadedecay	Tau: cascade decay

coordinate systems, we can be sure that the analysed figure contains at least one plot. This kind of reasoning could be encoded as derivation rules in the ontology. However, in some cases the description might be too complex, making the regular code written in a general purpose programming language a more viable option

4.5 Annotating Entities with External Vocabularies

Section 3.3.7 describes how figures extracted from scholarly publications encoded in PDF format are annotated with their top-level metadata using the HFO vocabulary. Previous sections of the present chapter describe the prototype of a tool analysing the content of figures and extracting the metadata of the coordinate systems present in the figure. However, there is a third type of annotation allowed by HFO, for which we have not provided a method of automatic metadata generation: the annotation with concepts from external ontologies and with units. This section delves into the method of discovering the references to concepts which are encoded in string annotations. The described algorithms for discovering concepts from HEP taxonomy and MUO units can be applied to most of the string annotations generated during the extraction of plots and during the extraction of the internal semantics.

Table 4.2 presents some of the concepts from the HEP taxonomy, together with their labels. In the case of basic concepts (those, which are not composed of sub-concepts), the label consists of a sequence of words. In the case of composite concepts, parts responsible for the partial concepts are separated using a colon. In many cases, the order of words which appear in the annotated text is different from the one present in the HEP taxonomy label. However, in order to establish a connection, we require all the words to appear in a contagious fragment.

The annotation begins with preprocessing all the labels present in the HEP ontology, in a way allowing us to efficiently search for arbitrary word permutations. First, every label (or alternative label) is split into a set of words. In this process, all the colons and white spaces are removed. Subsequently, every separate word is indexed (maintaining a link to the original label and the HEP concept).

Similarly, the text which is supposed to be annotated with the HEP concepts is split into a sequence of words. A search is performed using any continuous subsequence of the input. Every subsequence is treated as a set of words (Repetitions are considered only once and the order is ignored) and matched with possible labels (Also treated as a set of words). A match is established only if both sets are equal.

The matching algorithm attempts to find the most specific concepts. This is achieved by searching for the longest matching descriptions. If a sequence of input words matches a concept from the HEP taxonomy, no subsequence is checked. However, we allow other sub-sequences, which intersect only partially with a matched one, to be matched with HEP concepts.

A different method of matching was used for the units. The identifiers of units tend to be very short and easily confused with legitimate words of a natural language and potentially leading to incorrect matches. Moreover, in many cases, units are indicated by only a part of a word. Such a situation can occur when the unit is written together with the prefix, it appears after a numerical literal (with which it forms a word) or when unit is indicated by the presence of brackets.

At the first stage of the annotation process, the string which should be annotated is split into a sequence of words (substrings separated by a non-zero number of whitespaces). Subsequently, we select elements of the sequence, which have a potential of encoding units. This is done by selecting words which appear after a number, parts of words which start with a number, and parts of words which begin by the opening bracket and end by the closing one.

Each of the selected words was processed by searching for the longest suffix which matched with a label of some unit, making the remaining prefix match with some prefix label. The algorithm starts with a suffix consisting of the entire word and iteratively decreases its length, at the same time increasing the length of a potential prefix. In the case of unit units, we always match the entire labels and in a case-sensitive manner.

4.6 Summary

This chapter has described a method of automatically discovering the semantics of figures extracted using the method of Chapter 3. We presented how the semantics extraction problem can be divided into separate modules and then demonstrated a feasibility of implementing such modules by implementing an extraction module capable of detecting coordinate systems inside figures. We also demonstrated how the metadata produced using the extraction module could be encoded using the HFO ontology.

The implemented module is only a small part of a large system that enables the extraction of semantics from plots. Currently, the presented data model contains more concepts than what can be automatically described using the implemented prototypes of feature extractors. The future work in first place has to concentrate on extending the number of detected entities. In particular, the description of data-lines should be done. The structure of the HFO makes this extraction a much easier task than automatically extracting the data encoded by plots. Such a situation is possible because the description concentrates on the geometrical primitives seen from the point of view of the detected axes.

The result of this work will consist of a full figure annotation using the complete HFO ontology.

Additional work has to be also done in order to provide the annotation of diagrams. However, plots appear in the HEP publications much more frequently and as such, they should attract more attention.

Chapter 5

Searching for Figures

5.1 Introduction

Semantically describing figures and developing methods for their automatic annotation form only the preliminary step for the real applications. This chapter explores the methods of searching for figures.

There exist different approaches to searching. First, we describe the search for a particular know concept inside annotated figures. This type of search allows discovering figures which satisfy particular user-specified criteria. The search for the content encoded using the RDF triples has been tested by using a small working prototype.

Another approach to searching involves the usage of examples. Users can specify an example figure and the system should be able to retrieve figures which are similar. This type can be implemented using a method for determining how similar two graphical representations are. We have explored methods of constructing such similarity measures which take into account not only the visual similarity between pictures (which in the case of scientific figures might be deceiving), but the similarity of their semantic annotations.

The remainder of this chapter is structured as follows. In section 5.2 we describe the existing approaches to searching for figures and we present the description of the existing SPARQL Protocol and RDF Query Language (SPARQL) searching engines. Section 5.3.1 describes the environment which has been used for the execution of tests. Section 5.3.2 shows examples of different SPARQL queries and the analysis of their performance in comparison with a classical system using keyword-based indexing. Section 5.3.3 presents examples of more complex queries using the internal structure of the figures rather than their global annotation with concepts of physics. Section 5.4 describes the construction of a similarity measure based on the semantic annotation of figures. Later, in Chapter 6, we provide applications of the similarity measures for a real system managing figures.

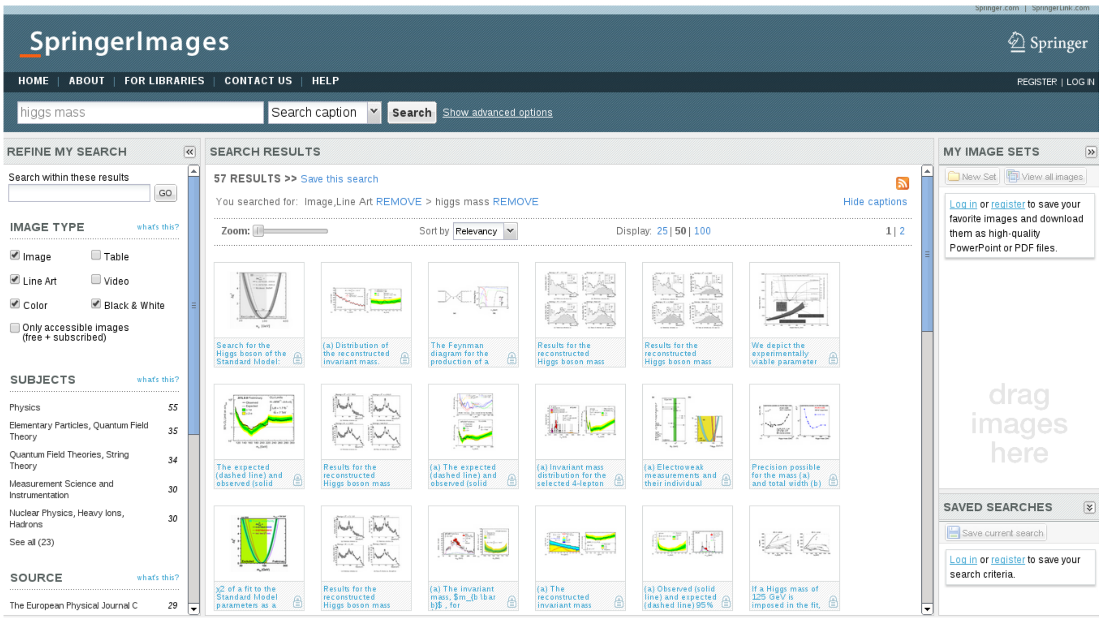

SpringerImages
Springer
HOME | ABOUT | FOR LIBRARIES | CONTACT US | HELP
REGISTER | LOG IN
higgs mass
Search caption
Search
Show advanced options
REFINE MY SEARCH
Search within these results
GO
IMAGE TYPE
Image
Table
Line Art
Video
Color
Black & White
Only accessible images (free + subscribed)
SUBJECTS
Physics 55
Elementary Particles, Quantum Field Theory 35
Quantum Field Theories, String Theory 34
Measurement Science and Instrumentation 30
Nuclear Physics, Heavy Ions, Hadrons 30
See all (23)
SOURCE
The European Physical Journal C 29
SEARCH RESULTS
57 RESULTS >> Save this search
You searched for: Image,Line Art REMOVE > higgs mass REMOVE
Hide captions
Zoom:
Sort by Relevancy
Display: 25 | 50 | 100
1 | 2
MY IMAGE SETS
New Set
View all images
Log in or register to save your favorite images and download them as high-quality PowerPoint or PDF files.
drag images here
SAVED SEARCHES
Save current search
Log in or register to save your search criteria.

Figure 5.1: The search page of Springer Images.

5.2 State of the Art

The search for images (figures in particular) is an important topic of research. Nowadays, there are an increasing number of systems providing some kind of functionality for figure searching. For instance, important scientific publishers like Springer or Elsevier have created services to allow access to figures present in scientific publications: the improvement of SciVerse Science Direct site for searching images in the case of Elsevier[1][22]; and the SpringerImages service in the case of Springer[2][21]. These services allow searches triggered from a text box, where the user can introduce a description of the required content. It is also possible to browse images by categories such as types of graphics (Image, Table, Line art, Video and so on). The search engines are limited to searches based on figure captions. In this sense, there is little difference between the image search and text search implemented in a typical digital library.

Figure 5.1 shows the search interface provided by Springer Images. The user is offered a possibility of searching for phrases appearing inside the captions and full text of images. Results of the search can be restricted to different general types of graphics which can be selected from the panel on the left side of the interface. Figure 5.2 shows which detailed information about a figure is displayed by the system.

[1]`http://www.sciencedirect.com/` (last access: 12.02.2013)
[2]`http://www.springerimages.com/` (last access: 19.08.2013)

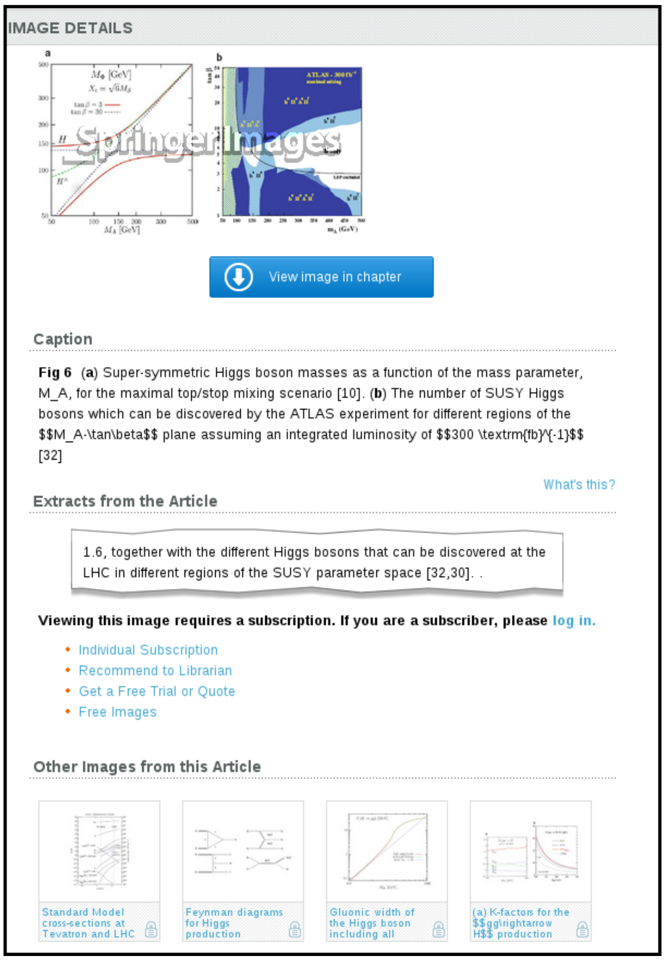


IMAGE DETAILS

Caption

Fig 6 (**a**) Super-symmetric Higgs boson masses as a function of the mass parameter, M_A, for the maximal top/stop mixing scenario [10]. (**b**) The number of SUSY Higgs bosons which can be discovered by the ATLAS experiment for different regions of the $$M_A-\tan\beta$$ plane assuming an integrated luminosity of $$300 \textrm{fb}^{-1}$$ [32]

What's this?

Extracts from the Article

1.6, together with the different Higgs bosons that can be discovered at the LHC in different regions of the SUSY parameter space [32,30]. .

Viewing this image requires a subscription. If you are a subscriber, please log in.

- Individual Subscription
- Recommend to Librarian
- Get a Free Trial or Quote
- Free Images

Other Images from this Article

Standard Model cross-sections at Tevatron and LHC

Feynman diagrams for Higgs production

Gluonic width of the Higgs boson including all

(a) K-factors for the $$gg\rightarrow H$$ production

Figure 5.2: The details of a figure displayed by Springer Images.

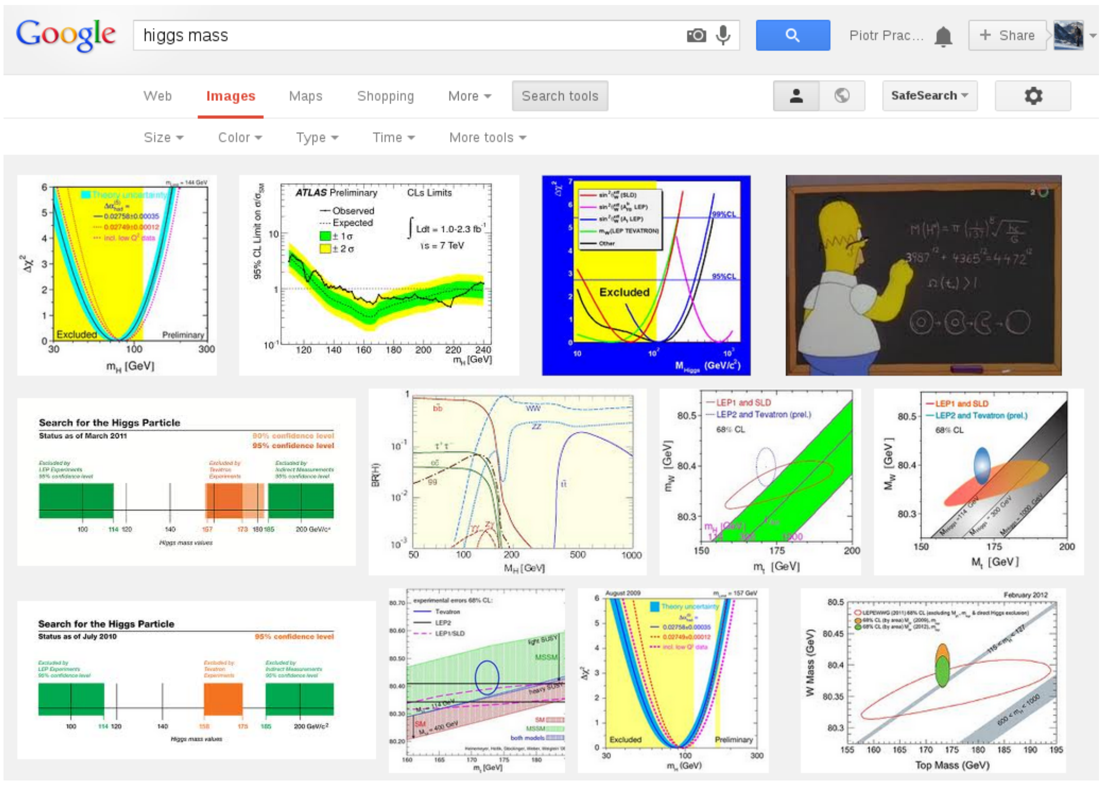


Figure 5.3: Searching for figures using Google Images.

Springer Images is an example of a service which is oriented towards the scholarly publications. A different approach has been implemented by Google Images [3] which is a general-purpose search engine. Similarly to Springer Images, Google Images allows users to search for figures based on their textual description inside the original document. It also allows searching by providing an example image. However, Google Images is oriented towards the general images rather than processing the scientific graphs in particular. The available categories (e.g. "clip art") are not optimised towards the management of scientific images. As such, the results are less precise comparing to the dedicated systems managing figures. Moreover, Google Images provides only the search engine and relies on the external storage of graphics. In many cases, the results of Google Image search point to the specialised systems.

There exist multiple approaches to searching for the content. Most of the state of the art search engines use one of the techniques of indexing text. In most of the cases, the scenario is similar: the collection of documents which should be made searchable is preprocessed and search indices are created. These indices are later used to quickly retrieve the relevant documents without analysing their content. A typical search query consists of a word or a phrase. The results consist of all the documents which contain this word or phrase. The indexing techniques are in many cases enhanced with preliminary Natural Language Processing (NLP) which aims at normalising the data. For example, words can be replaced with their basic grammatical forms or roots. Many digital repository software suites implement their own search engines. Last years have also witnessed the development of general purpose libraries providing efficient search engines (Lucene[4], Solr[5]).

Text-based search is a powerful tool allowing the discovery of the textual documents and with necessary extensions, also other objects. However, in its general form, it fails to describe the connections between searched concepts. In recent years, much effort has been devoted to developing the methods of searching using the semantic annotation. SPARQL[3] allows searching for objects annotated using semantic technologies (RDF). SPARQL queries have a form of graph patterns which are matched against fragments of the semantic annotation. In the case of annotation with literal values, a number of matching and comparison operations allow the classification of entities as relevant or irrelevant. SPARQL is implemented in many libraries and software suites managing the semantic annotation (Jena[6], Protege[7]). The efficient execution of SPARQL queries in both single-machine and in the distributed environment is object of active research [28].

The semantic annotation can also be used to compare entities. There exist well studied approaches to comparing semantic annotations. In particular, Pequita et al. [40] provide an

[3]`http://images.google.com` (last access: 19.08.2013)
[4]`http://lucene.apache.org/core/` (last access: 20.08.2013)
[5]`http://lucene.apache.org/solr/` (last access: 20.08.2013)
[6]`http://jena.apache.org/` (last access: 20.08.2013)
[7]`http://protege.stanford.edu/` (last access: 20.08.2013)

overview of the most common methods. In this chapter we present also a method of constructing a similarity measure comparing semantic annotation which is based on the methods described in this work. However, the hierarchical structure of the annotation is more prominent in our application than in the work of Pequita and we extend the presented methods to take advantage of this fact.

5.3 Direct Searching Based on Semantic Technologies

5.3.1 The Testing Environment for Semantic Searching

Before being able to perform any experiments concerning searching for figures, we needed to prepare a database of semantic descriptions and populate it with sample figures and their description. In our experiments we used Apache Jena library. We randomly selected a set of 76 publications stored in INSPIRE. All those articles were subsequently processed in the automatic extraction process, which produced the metadata of figures together with their vector graphics representations. In order to be able to evaluate the search system, we wanted to have the knowledge base of the highest possible quality. We manually reviewed all the extracted results, rejecting few which were extracted incorrectly. Subsequently, we performed the automatic annotation on the set of obtained results. Outcomes were uploaded into the knowledge base (in the stage of experiments, stored in an RDF file). After all the preliminary processing, out of 76 processed publications, 59 contained at least one figure or table. In total, we gathered descriptions of 517 figures and tables.

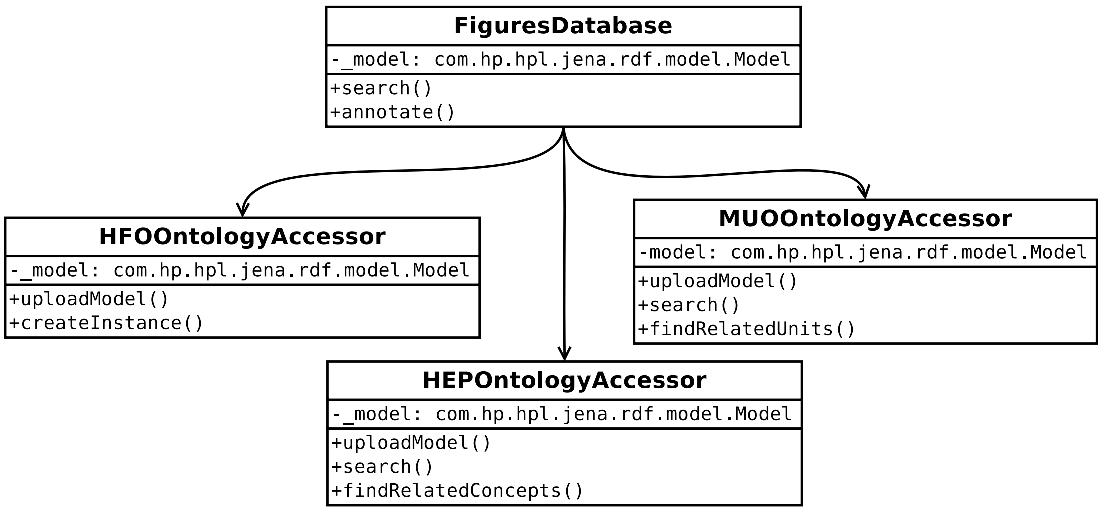


Figure 5.4: The internal architecture of the prototype used to search for figures using their semantic annotation in RDF.

Once the knowledge base was created, we developed a search component to facilitate the translation of user queries into SPARQL queries. Figure 5.4 shows the internal architecture of the component allowing the management the semantic annotation of figures and to execute searches. The main class (*FiguresDatabase*) provides interface to manipulate the sorted annotation. It uses a number of auxiliary classes (*HEPOntologyAccessor*, *MUOOntologyAccessor* and *HFOOntologyAccessor*) to manage the concepts of the used ontologies and to retrieve information concerning their concepts.

The standard way of accessing the data encoded in the form of semantic triples is SPARQL. A typical SPARQL query retrieves a set of individuals that satisfy certain criteria. The criteria can specify certain structural properties of the graph or refer to the properties of the literals (interpreting them as numbers or strings and performing standard operations on them).

The results retrieved using SPARQL queries can consist of arbitrary individuals described using RDF, however in our present work we restrict our attention to retrieving only entire figures rather than their constituent parts. A figure is the central concept of our prototype and will be treated as a first-class artefact also in the constructed information system.

In traditional digital library systems, the central unit of information is a publication described by a bibliographical record. The integrated search engines typically operate on records allowing their efficient retrieval. Our long-term goal is to integrate the described figure search with a digital library system. Our more holistic approach to figure search makes the integration easier at the conceptual level. A more detailed description of the integration with the INSPIRE search engine can be found in Chapter 6.

5.3.2 Testing Concept-based Queries

This section presents sample SPARQL queries which can be executed on the knowledge base which we have constructed. We present queries of different complexity, having a potential of being useful in a scientific document repository.

We begin with a very basic example of retrieving objects of a given type. Listing 5.1 shows a query allowing the retrieval of all plots. This type of searches is fundamental to the figures retrieval. However, it is not difficult to simulate a similar search using the internal search engine of INSPIRE. Moreover, the recall and precision of this query executed on our knowledge base is always 1, because we manually reviewed the results of the figures extraction and the decision if an object is a table or a figure is made at exactly this stage.

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX hfo: <http://www.semanticweb.org/ontologies/invenio/inveniomodel.
    owl#>
SELECT ?figure
WHERE {
   ?figure rdf:type hfo:Plot
}
```

Listing 5.1: A query retrieving figures of a given type

A similar query can be used to retrieve all figures, which have been annotated with a particular concept from the HEP ontology. The query from Listing 5.2 can also be simulated using the search system of a traditional digital library that allows searching records about a topic. However, a more flexible generalisation of this query would be more difficult to simulate.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX hep: <http://cern.ch/thesauri/HEPontology.rdf#>
SELECT ?subject
WHERE {
   ?subject dc:subject hep:hadron
}
```

Listing 5.2: Searching for figures annotated with a HEP concept

The presented query searches only for the annotation of the main figure entity. However, the data model of HFO allows to link different constituent parts of a figure with the HEP concepts. SPARQL is a language allowing the query of the graph structure of the RDF description. A simple modification of the query depicted in Listing 5.2 allows finding annotations of any elements of a figure.

As previously mentioned, when annotating with individuals of the HEP ontology, we were trying to describe the most specific terms (In the linguistic sense - defined by the *skos:narrower* relation). HEPOntology contains also a thesaurus of concepts of HEP. In many cases, user might search for a term which is general. However, some figures relevant to the query might be annotated with more specific concepts. Without extensive data preprocessing, a basic search based on the exact matching of metadata values would not be able to exploit the dependency between the concepts.

```
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX hep: <http://cern.ch/thesauri/HEPontology.rdf#>
SELECT ?subject
WHERE {
   ?subject hfo:contains* ?s1.
   ?s1 dc:subject ?concept.
   ?concept skos:broader* hep:hadron
}
```

Listing 5.3: A more complex version of the query searching for annotation with a HEP concept

Listing 5.3 presents a more complex version of the query from the Listing 5.2. In this case, the searched concept is required to be a specification of the original concept from the query. Moreover, it can describe a part of the result figure.

We used the previously created knowledge base in order to evaluate, the impact of replacing standard exact-match queries with hierarchy-aware searches. The standard manner of evaluating the quality of search results involves the usage of the concepts of recall and precision. Precision is the ratio between the number of relevant entities which have been retrieved and the number of all retrieved entities. Recall is the ratio between all the relevant retrieved entities and the number of relevant entities. The execution of present test had the objective of measuring the quality of searching within a correctly extracted knowledge base. We also interfered with the extraction process by eliminating the misextracted figures. As a consequence, we assume that the knowledge base is 100% correct. This assumption is very similar to assuming that the errors are spread uniformly across the entire database of figures (which would be more realistic for the production system). For this reason, our assumption is a good estimate for the results which we could obtain on the entire set of samples from INSPIRE.

Assuming a complete correctness of the annotated samples set has a profound influence on the interpretation of results. In such a setting, calculating the precision always results in the value 1. This is because all samples are annotated correctly and there is no chance of retrieving an unrelated entity. Similarly, searching for all entities related to the concepts from the hierarchy of the desired term, can be considered as retrieving all the correct answers and no incorrect. This allows defining the recall of the exact matching query as the number of returned results divided by the number of results retrieved using the extended query.

In both examples from Listing 5.2 and Listing 5.3, we are searching for figures annotated with the concept of hadron. Hadrons are a wide class of particles, which share the common property of being constructed from quarks. There are many different types of hadrons (e.g. mesons, protons, antiprotons). All of these are relevant when searching for hadrons. The first query, simulating the search which could be achieved using exact-match indexing of INSPIRE, returns 4 results. The modified query, exploiting the dependencies between concepts, finds 10 results. This means that the recall of the exact query is in this case equal to 0.4. Similarly, searching for leptons returned 28 and 37 results. This corresponds to the recall of 0.76 for the first query.

The recall of the exact match query can vary from 0 in the case of searching for a general term, which is not directly used in any annotation to 1 for the case of searching for the most specific term of the taxonomy. Table 5.1 summarises the results we have obtained by executing the evaluation over different sets of HEP concepts. The results presented in the table represent arithmetic averages of recalls for all the concepts for which at least one result has been found by the more general query.

We performed the experiment of comparing the number of returned results for all the HEP

Table 5.1: Increase of recall of the semantic search prototype in comparison with the INSPIRE system

Query	**exact-match query**	**Semantic search prototype**
All the HEP concepts used to annotate at least a single figure	0.984	1
All the concepts for which there exists a narrower term in the HEP ontology	0.915	1
All the concept from the HEP ontology	0.848	1

terms used to annotate at least one figure. These results include all the specific concepts which we have used for the annotation and ignore the existence of the concepts which would not give any results during the exact matching, while yielding results when executing the general query.

We repeated the experiment with all the used concepts which could be further specified using the HEP taxonomy. The result was lower. The last row of the table represents results which we have obtained when computing the recall using all the concepts from the entire HEP ontology. The last number is expected to be higher than the one obtainable from a larger knowledge base as the number of annotations is lower in the case of our testing set.

In all cases, the usage of a more complex query, which can be executed because of the usage of a RDF database, increases the accuracy of retrieved results.

5.3.3 Testing Complex Queries Restricting Figure Internals

The prototype of automatic figure annotation allows describing only part of the attributes of a figure predicted by the HFO data model. More work is needed to implement an automatic extraction of data lines and other more complex properties. Even though the complete extraction lay outside the scope of the thesis, we needed a method of evaluating the relevance of the data model which we propose.

We extended the automatically-generated database with 3 individuals, which we have described in a more thorough manner. Later, we attempted the execution of specialised SPARQL queries designed to retrieve the annotated individuals from among the entire database.

As aforementioned, the automatic annotation process was at the moment of writing unable to produce a comprehensive description using all the concepts of the ontology. Producing a complete description of the data lines encoded in a figure is a laborious, largely mechanical process. However we wanted to be able to prove that the model can be used for the data retrieval. We manually described a set of 3 different figures, which allow us to construct a basic proof of the feasibility of the system.

Our first example is depicted in Figure 5.5. The presented plot depicts measurements of the same quantity performed by different experiments. We manually created a description of

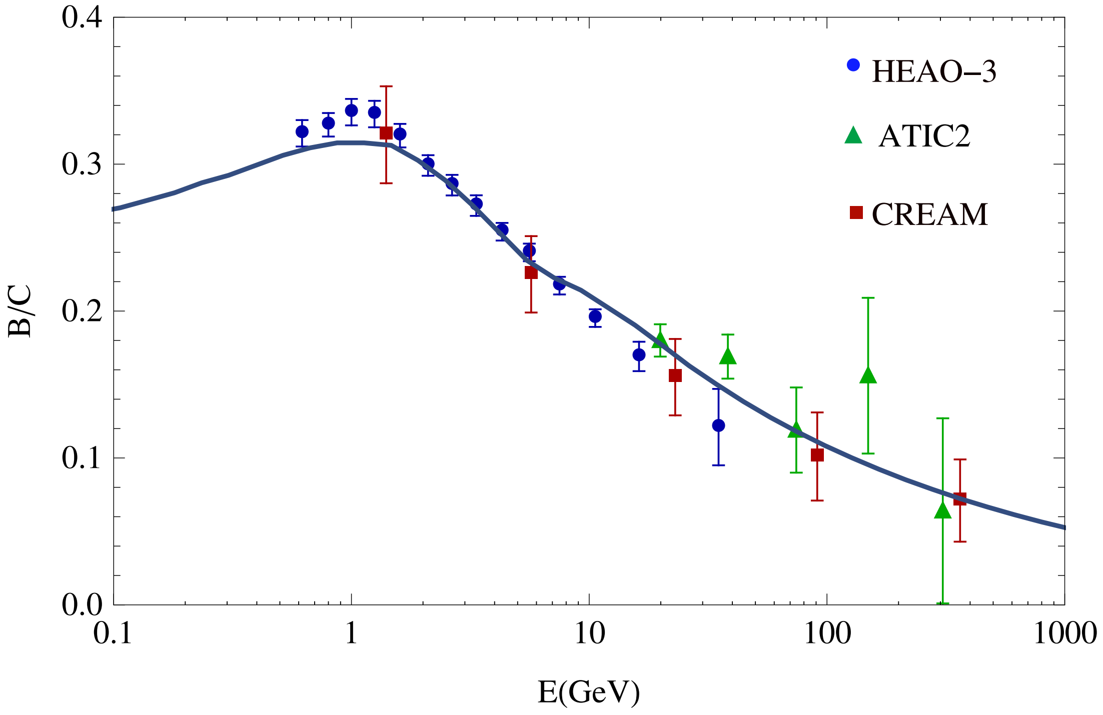


Figure 5.5: An example of a figure injected into the automatically-generated knowledge base

the datalines present in this plot and incorporated it in our knowledge base.

The query which we tested using this example can be seen in Listing 5.4. We wanted to retrieve all the plots which contain datasets which have been annotated with the Cosmic Ray Energetics And Mass (CREAM) experiment (which is represented by an individual from the HEP taxonomy).

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX hfo: <http://www.semanticweb.org/ontologies/invenio/inveniomodel.
    owl#>
PREFIX dc: <http://purl.org/dc/elements/1.1/>
PREFIX hep: <http://cern.ch/thesauri/HEPontology.rdf#>
SELECT ?subject
WHERE {
   ?subject rdf:type hfo:Plot.
   ?subject hfo:contains* ?ds.
   ?ds rdf:type hfo:DataSeries.
   ?ds dc:subject hep:CREAM
}
```

Listing 5.4: A query searching for a plot which contains a data series related to the CREAM experiment

When executing over our knowledge base, the query has proven to return correct results (In this case, consisting of a single plot).

Figure 5.6 shows a blue-band plot, which we have added to the database. Blue-band plots are an example of encoding meaningful physics properties using colours in a plot (typically, the choice of colours is part of the graphical design and does not carry any meaning).

Listing 5.5 shows an example of a more complex query, which can be used to search for figures containing a blue-band plot. The criterion of being blue is expressed as having the blue component with a higher numerical value than other colour components. In addition, the plot has to contain an axis which is related with a unit derived or equivalent to electronvolt.

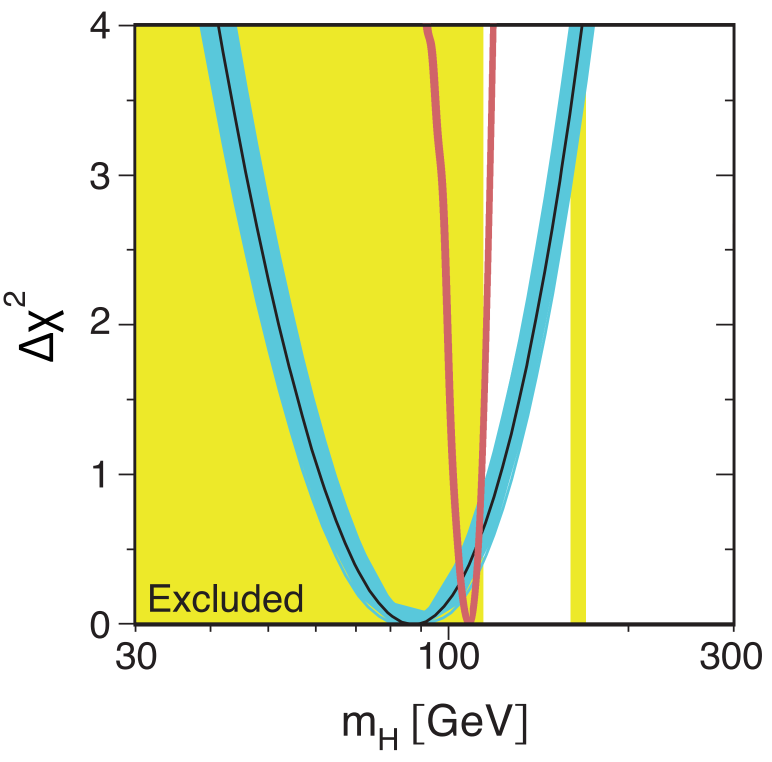


Figure 5.6: Example of a blue-band plot popular in HEP

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX hfo: <http://invenio-software.org/hepfigures#>
PREFIX colours: <http://kaiko.getalp.org/kaiko/ontology/colors.owl#>
PREFIX xsd: <http://www.w3.org/2001/XMLSchema#>
PREFIX muo: <http://purl.oclc.org/NET/muo/muo#>

SELECT ?plot
WHERE {
    ?plot rdf:type hfo:Plot.
    ?plot hfo:contains ?dataseries.
    ?dataseries rdf:type hepfigures:DataSeries.
    ?dataseries hfo:contains ?shape.
    ?shape rdf:type hepfigures:DataArea.
    ?shape hfo:hasFillColour ?colour.
    ?colour rdf:type colours:COLOR.
    ?colour colours:hasBLUE ?blue.
    ?colour colours:hasGREEN ?green.
    ?colour colours:hasRED ?red.
    ?dataseries hfo:hasArgumentAxis ?axis.
    ?axis hfo:hasUnit ?unit.
    ?unit muo:derivesFrom* <http://purl.oclc.org/NET/muo/ucum/unit/
       energy/electronvolt>.
    FILTER xsd:integer(?blue) > xsd:integer(?green) &&
           xsd:integer(?blue) > xsd:integer(?red).
}
```

Listing 5.5: A query retrieving candidates of blue-band plots

5.4 Similarity Measures Based on the Semantic Annotation

Having figures annotated in a semantic manner allows much wider applications than only searching in a manner which takes into account the hierarchy of the notions. One of such applications is discovering the connections and similarities between different figures. The classical approach of discovering similarities between objects uses the notion of a similarity measure or a distance. The similarity measure is a function which takes two arguments and returns a single number. The number has to be greater or equal to 0 for all possible pairs of arguments. The 0 value should be attained if and only if both arguments are the same. In a strictly mathematical sense, it should also satisfy the triangle inequality. However, in many practical approaches related to the artificial intelligence, the latter requirement can be relaxed. Moreover, the value of the distance function should remain the same after inverting the order of applied arguments. For any two objects, the value of the similarity function indicates how close, in the sense of this function, the arguments are. The most obvious examples include the distance in a 2 or 3 dimensional spaces. More abstract distances are used throughout mathematics and in applications of the Artificial Intelligence.

In the case of images, similarity measures can be for example used to compare and detect people in pictures. There are a number of approaches to measuring the similarity between images. The most typical are based on comparing certain features of the pixel representations.

These methods, however efficient for pictures, do not have a great potential for being effective in the case of scientific figures. Most of the known methods tend to emphasise the visual similarities, while two plots looking almost the same could contain completely different content. On the other hand, the semantic annotation contains all the data which is important for the interpretation (and thus, also the comparison) of figures. This section explores the possible methods of constructing semantics-aware similarity measures for plots. Additional work needs to be done to design a measure which could be used to compare diagrams.

5.4.1 The Structure of the Similarity Measure

The semantic annotation, as described in Chapter 2, is hierarchical. Figures can contain plots (as a special type of a subfigure) which in turn contain axes which contain ticks. The way in which elements can be combined is strictly defined and cannot be changed. For example, it does not make sense to include an axis tick as a direct element of a figure. The similarity measure for figures (and plots in particular), should inherit this hierarchy. Our proposed similarity measure for plots compares different levels of the description and combines these values to obtain the final score.

The hierarchical structure of the comparison method can be illustrated using the example of comparing two figures which contain subfigures. In this case, we should compare all the pairs of subfigures which contain a subfigure from one and the other figure. For example if the first figure (f) contains subfigures $f_1, f_2, \cdots, f_N$ and the second figure (g) contains subfigures $g_1, \cdots, g_M$, the distance between figures f and g could be expressed using the formula:

$$d(f,g) = min_{i \in \{1..N\}, j \in \{1..M\}} d(f_i, g_j)$$

In this case, the total distance is defined by the pair of subfigures, which is the closest to each other. Such a definition seems to be intuitive for the case of searching for a figure which contains information the closest to the given one. However, different scenarios could require different composition of results from the subfigures. A technique similar to the one presented in the state of the art can be used to compute distance between different composite elements of the annotation such as coordinate systems or axes.

The above method of comparing is suitable for sub-elements of the same type. However, a larger entity can contain different types of annotations. For example a plot can contain coordinate systems, but also caption or the full text. In such a case, only annotations of a similar type should be compared. The results of those comparisons can be combined using a system of weight coefficients.

If we want to compare concepts c and d which are annotated using predicates $p_1, ..., p_N$, we can define the distance between c and d in the following manner:

$$d(c,e) = \sum_{i=i}^{N} C_i d(\{x : c\ p_i\ x\}, \{x : e\ p_i\ y\})$$

where the distance between the sets of elements can be defined in a similar manner as earlier for the subfigures. Additionally, the coefficients should sum to 1

$$\sum_{i=1}^{N} C_i = 1$$

The role of the weight coefficients is to emphasise the importance of certain metadata fields and to decrease the importance of the others. A trivial example of weighting the annotation terms is rejecting the terms responsible for the material annotation of figures. Those elements of annotation do not carry important meaning and should be completely omitted when calculating the distance.

There is an important property of the annotation which should be noted at this point. In many cases, one of the compared annotations might be missing certain annotation elements. For example, a figure might not have the caption text. When comparing such a figure with a figure containing a caption, the distance between the set of captions of one and the other would be calculated. In such a situation one of these sets will be empty. The distance should not be in such a case considered to be infinite. There should either be a default value for comparing with empty sets or the coefficient corresponding to the comparison of captions should be modified to 0 and other coefficients should be increased so that their total sum would remain 1.

When comparing figures, we have to compare a number of direct annotations (with title, the complete text, caption and so on ...). We also have to consider other elements of the figure (like coordinate systems for plots). The following sections describe how to compare the direct annotations and coordinate systems and data lines. The coefficients allowing the combination of those results should be determined by the way of experiments with the corpus of annotated data.

5.4.2 Similarity of Coordinate Systems

The main features which should be considered when comparing the coordinate systems are the quantities which are compared and the units. The quantities displayed in a coordinate system should have the highest weight when calculating the overall distance between two of them. Two coordinate systems describing different physical phenomena should not be considered very similar even if the units and displayed values are similar.

The usage of the MUO ontology allows a much more robust description of the used units. In the case of composite units, their base units should have a much higher weight than the modifiers.

Comparing axes should also consider the values of axis tick labels. Those labels usually encode numbers. When calculating a distance between two coordinate systems, the system should try to interpret values of those labels. If the detected labels can be interpreted as numbers, the system could assess the value ranges encoded by the system. In the process of the interpretation, the system should consider prefixes attached to the composite units which describe axes. Decoded numerical values should be multiplied by numerical constants related to those prefixes (1'000'000 for "mega" and so on . . .). A total similarity of numerical ranges could be assessed by considering two value intervals and then calculating the size of their intersection. The obtained number should be normalised using the size of a sum of both intervals.

In many cases, axis tick labels encode numbers, but it is impossible to decode them using standard methods. This typically happens when a symbolic representation is for some reasons more readable for a human reader (for example 10^6 instead of 1'000'000). Many of those annotations could be automatically interpreted (there exist complex computer systems allowing symbolic calculations). However, if such an interpretation is impossible, coordinate systems should be compared in the same way as if axis tick labels were not present at all.

There are many features which should not play any role when comparing the coordinate systems. For example, the number and type of axis ticks should not play any role.

5.4.3 Similarity of Data Lines

The objective of data lines is to encode functions. The most effective method of comparing data lines should involve the knowledge of the values of the function in different points. However, as described in Chapter 2, we wanted to avoid back engineering the exact numerical values and binding them with the exact points. If a future extension to the HFO and to the annotation of figures allows discovering and describing the exact values, the system could use one of the methods of calculating a distance between functions. Such methods are well known in the mathematics discipline. In order to calculate such a distance, one could for example use the norm of the L_2 space:

$$d(f,g) = ||f - g|| = \int_D (f(x) - g(x))^2 dx$$

Such an integral should be calculated over the intersection of the described domains and be normalised using the size of this domain.

At present, we cannot design a similarity measure comparing datalines being so precise. However, we can try to compare the shape of the plots. Data lines described using HFO are encoded relative to the length of the coordinate system axes. The similarity measure should attempt to use these values. First, data lines could be aligned (shifted in vertical and in horizontal directions to minimise the summary difference between them). Subsequently, a distance similar to the one from the above formula should be calculated.

Plots tend to contain many data lines. The total distance between two sets of data lines coming from different plots can be calculated using the pair of data lines having the smallest distance. The accuracy of the above method is low because of using coordinates relative to the length of the axes. This is why the weight coefficient connected with the data lines similarity should be relatively low.

5.4.4 Similarity of Term Annotations

In order to compare figures, at the lowest level we need a method of comparing term annotations. By term annotations we understand predicates which have which is a literal or an individual defined in some ontology.

The meaning of such annotations can be standalone or dependent on the annotation of other elements. An example of a standalone annotation is the full text of a figure or the associated concept in the HEP ontology. The meaning of the text annotating a tick label is on the other hand dependent on the annotation of the axis to which the tick belongs.

This section focuses mainly on the comparison of standalone annotations, however, similar techniques can be in some cases used to compare more hierarchy-dependent term-annotations.

The comparison of literals can be procssed using standard value-comparison techniques. For numbers, it is possible to use the numerical distance between them and for texts, we can use the edit distance. There also exist standard techniques allowing the comparison of concepts. In principle, they can be divided into two main categories: comparison based on edges and comparison based on nodes.

Methods which compare concepts based on edges attempt to find paths connecting concepts inside the ontology. These methods differ in the way, how they transform the discovered paths into numerical values. A particular methodology should be adjusted to the characteristics of the problem to which a particular similarity measure should be applied. The role of edges inside the ontology graph is served by the predicates. The similar assessment method could search for paths passing through particular types of predicates (for example, in the case of SKOS-based knowledge systems, we could consider generalisations of terms). Having found all the paths, the method can take into account all or only some of them (for example by calculating an average between all the paths, by taking the longest or the shortest path). The evaluation of a single path can also be performed in different ways. In a simple scenario, the algorithm can calculate the number of edges belonging to the path. However, there are also more complex possibilities. For example, edges can have different weights.

A completely different class of approaches tries to exploit properties of particular nodes rather than only paths between them. These methods take advantage of the methods of the Shannon [49] Information Theory and are based on the concept of the Information Content (IC) of a node. Evaluating the IC requires all the nodes to be annotated with the probabilities of their appearance inside an annotation. If the concept c is used in annotation with the probability $p(c)$, then the IC of this concept can be expressed using the formula:

$$IC(c) = -\log_2 p(c)$$

Calculating the necessary probabilities can be done using a sample corpus of annotations. In such a corpus we can calculate the number of appearances of a given term and then we can assume that the probability that this term will appear in any annotation can be estimated by the calculated number of appearances divided by the number of investigated annotations. If we change the corpus, the probabilities are likely to change. However, for sufficiently large and representative corpuses, those values should be comparable.

There are multiple methods to compute the distance between concepts using the Information Content. However, a recurring schema involves finding common ancestors of two compared concepts and then calculating the information content of those ancestors. A more detailed description of different methods of comparing terms in an ontology, together with concrete formulæthat allow calculating those similarities, can be found in [40].

In general, the methods exploiting the path-distance inside an ontology tend to work better for ontologies where term definitions have comparable depth. Information Theory based methods do not make this type of assumption. Moreover, node-based methods tend to emphasise the importance of terms which are rare in the description. Less frequent concepts carry much higher information content than those which appear more frequently. This can be seen for example in the natural language, where the word “the” appears very frequently and carries very little meaning. On the other hand, the word “computer” appears much less frequently and thus it gives much more information to the reader.

The choice of a concrete similarity measure for term annotations should be determined using the experiments and the corpus of semantic annotations of the figures from INSPIRE. However, node-based methods seem to have a much higher potential of being useful for the case of comparing figures in HEP Ontology.

5.5 Summary

This chapter described the infrastructure which allows searching for figures annotated in a semantic manner. We presented examples of useful SPARQL queries allowing the retrieval of figures based on the physics concepts with which they have been annotated. The tests on a sample corpus of annotated data have shown an increase in recalled. Thanks to the use of a hierarchy of classes in the ontology, the number of relevant entries retrieved is higher.

We also presented more complex queries exploiting the annotated structure of figures. The prototype of Chapter 4 forms an important step towards automatic annotation of figures. However it does not provide a complete annotation for the creation of a large dataset which could be used to test the presented queries. The usage of a smaller testing set has shown that the proposed annotation allows retrieving relevant results and rejecting the others.

At the end of the chapter we presented a method of constructing a semantic similarity measure which can be used to determine which figures are similar. The described similarity measure has not yet been implemented. Additional work is necessary to implement it and evaluate the relevance of the results obtained by comparing figures using the measure. In the future, the presented similarity measure can be extended to allow the comparison of different types of objects.

This chapter is only the beginning of the work which needs to be done to provide usable figure search. There are multiple directions in which the present work can be continued. The most urging extensions involve the integration of the INSPIRE search engine with a semantic engine. Chapter 6 describes some aspects of this work. Additionally, after the system is available to the wide public, INSPIRE can be used to gather statistics of usage of the figure search. This in turn should lead to the identification of additional queries and to making them much more prominent.

Chapter 6

Applicability to a Real Project: the INSPIRE Project

6.1 Introduction

The methods described in previous chapters allow processing figures and the documents of publications. However, additional elements are needed to construct a system capable of serving figures to the users. The main incentive for our work was the possibility of using the results as a part of the information system for HEP. This chapter presents how the described methods of processing figures can be used in the environment of a digital library. The integration is illustrated with the example of the Invenio digital library software platform and INSPIRE (the digital library of HEP).

In many cases, figures (and especially plots) are graphical representations of datasets. In recent years, those datasets have also become object of interest of various data preservation initiatives. The results from the Parse.Insight survey 2007 [27] indicated that the HEP community needs a "neutral" preservation platform for research datasets. INSPIRE moves in the direction of becoming a preservation platform that allows accessing different publishing artefacts in the area of HEP. This chapter describes also the additional work of integrating INSPIRE with one of the dataset preservation initiatives and thus, making datasets the first class citizens of the information ecosystem. Introducing datasets to INSPIRE opened new possibilities of interlinking different types of bibliographical objects. In the context of integrating figures inside INSPIRE, it for example allowed linking a figure with the underlying dataset. Maintaining and displaying such links facilitates the reuse of the scientific results.

The research datasets integrated in INSPIRE have been collected over 25 years as a part of the HepData Project[1] funded by the STFC (UK Science and Technology Facilities Council). HepData compiles and makes easily accessible a comprehensive and up to date database of

[1] `http://www.ippp.dur.ac.uk/Research/Projects/HEPDATA.html`

the HEP experimental scattering data – total and differential cross sections, fragmentation functions, structure functions and polarisation spin measurements from a wide range of particle physics interactions at the colliders and the fixed target facilities world-wide. As well as a general user searchable interface to the database, HepData also provides a variety of 'data reviews' providing indexed data on a variety of topics. Most of datasets hosted by HepData are related to figures or tables present in the publications.

This chapter is structured in sections discussing different aspects of the integration. In Section 6.2 we provide a short overview of the architecture of Invenio and of the INSPIRE project, necessary to understand the remainder of this chapter. Section 6.3 describes the extensions to the document-storage model of Invenio, which allowed us to store figures together with their metadata. Section 6.4 provides a description of integrating the figure extraction method within the data ingestion workflows of INSPIRE. In previous chapters we gave examples of semantic searches allowing the retrieval of figures. Section 6.5 describes the integration of certain types of semantic searches with the Invenio search engine. A complete ecosystem of scientific work and scholarly publishing contains other types of entities which are related to figures. Section 6.6 describes the integration with the HepData system, which provided INSPIRE with the description of datasets. These datasets are an example of objects which can be linked with corresponding figures or tables. Section 6.7 shows possible directions of the continuation of the integration of figures inside INSPIRE.

6.2 INSPIRE

6.2.1 The Overview of the Architecture of Invenio

Invenio is a software platform originally created to serve as a basis for CERN Document Server (CDS)[2], now used in multiple repositories and digital libraries. Examples include the EPFL Infoscience portal[3], the Digital Repository of the University of Zaragoza[4] and soon the Astrophysical Data System (ADS)[5]. Written in Python with minor parts in other languages (like C and Javascript), Invenio runs under the control of the Linux operating system, using the Mysql, Apache - modwsgi stack. At the implementation level, Invenio uses the Jinja templating system together with the Flask microframework. The object-relational mapping is performed using the SQL Alchemy toolkit. At the moment of writing, records are stored in MAchine-Readable Cataloging (MARC)21[6] format, however, there exist ongoing projects aiming at making Invenio more format-agnostic. Invenio is divided into a number of modules implementing specific functionalities necessary in a digital library. Figure 6.1 shows an overview

[2]`http://cdsweb.cern.ch`
[3]`http://infoscience.epfl.ch`
[4]`http://zaguan.unizar.es/`
[5]`http://adsabs.harvard.edu/index.html`
[6]`http://www.loc.gov/marc/bibliographic`

of Invenio modules and their location in different application layers.

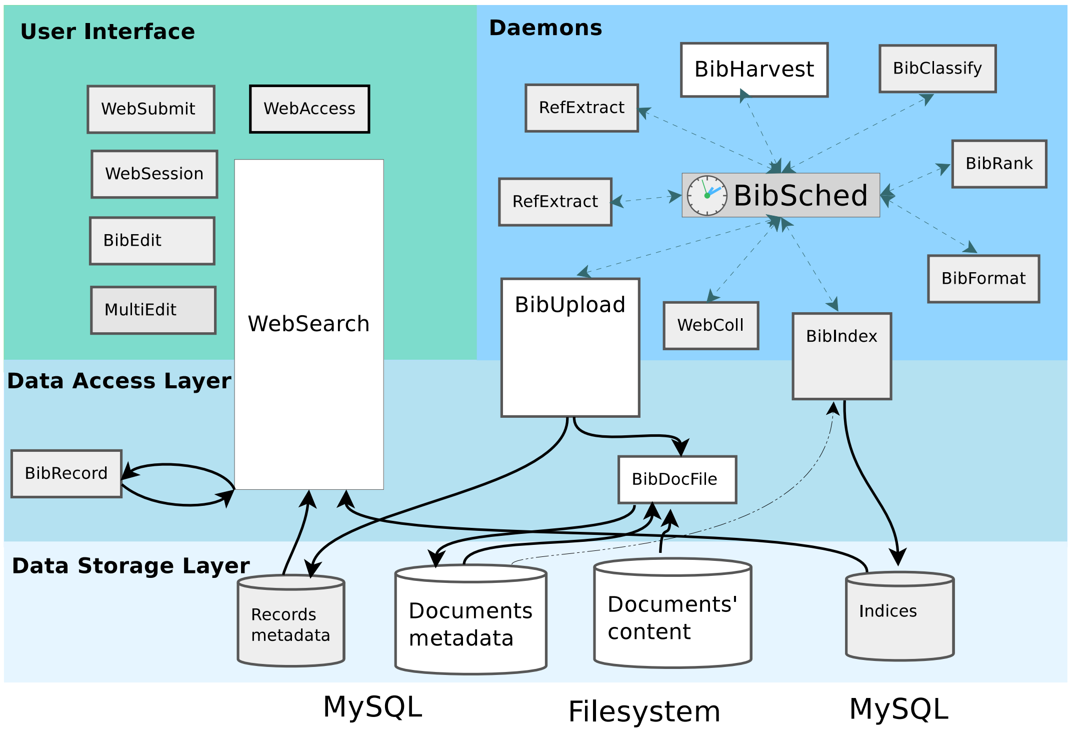


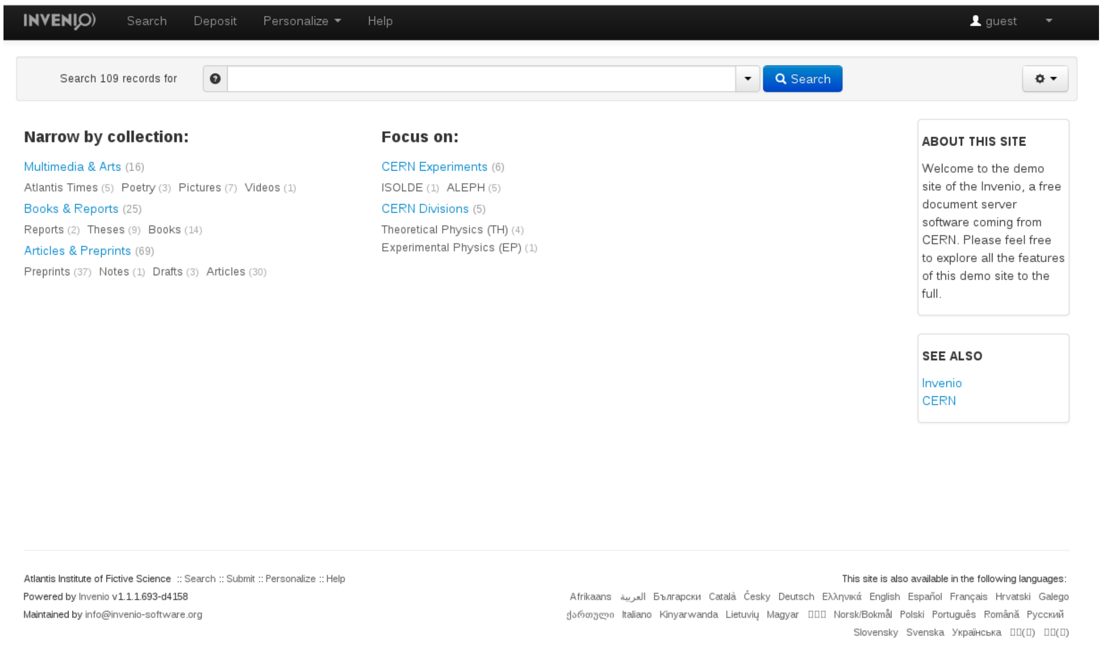
Figure 6.1: Overview of the architecture of Invenio. Modules marked with white background and a larger font are directly affected by our work.

Accessing the Content

Providing access to the stored data is the central functionality of every document repository. In the case of larger collections, it is important to provide a good method of finding the document of interest, for example using a search engine. The functionalities of providing access to the bibliographical metadata and of searching using previously calculated indices are implemented inside the *WebSearch* module. Every record stored in Invenio is identified by a number, which is unique throughout the entire installation. *WebSearch* provides methods allowing the retrieval of the metadata of a document identified by the specified number (In a format like MARC XML, HTML etc...). In every case the result is encoded as a single string value ready to be served to the users. *BibRecord* is the Invenio module which provides in-memory representation of records, which is more suitable for manipulations and for accessing particular metadata fields.

WebSearch also implements functions to perform searches across the corpus of the stored bibliographic objects. These functions accept a search query in an internal query language to search for keywords inside metadata fields (with recent extension, also in full text of files). The internal query language permits the usage of logic operations to combine results of different subqueries. It also allows the usage of regular expressions and wildcards. *WebSearch* splits the provided search queries into basic parts and uses the precalculated indices in order to retrieve the identifiers of records which satisfy the query. The basic queries are those, whose resolution requires lookup in a single (meta)data field. Subsequently to retrieving the record identifiers, the results from different elementary searches are combined to construct the final answer.

Accessing the data is only a part of the functionalities provided to the end users. A number of additional non-core modules (many of these can be disabled in a particular installation, like INSPIRE) enrich the user experience. All of them are based on the data-accessing capabilities of *WebSearch* and the underlying data model. Some of them also communicate with the subsystem scheduling the execution of back-end tasks. The user-experience related modules include: *WebAccess* (controls, who can access which content), *WebStyle* (manages the display style of Invenio pages), *WebAlert* (informs users when new interesting for them materials appear), *WebSubmit* (allows submitting new documents for the inclusion in the collection), and *WebComment* (allows commenting the managed documents). Figure 6.2 shows the main page of a default Invenio installation, featuring the search box allowing a free-text search for bibliographic records. It also shows the tree of document collections. Figure 6.3 shows an Invenio page displaying a particular bibliographic record.



Narrow by collection:

Multimedia & Arts (16)
Atlantis Times (5) Poetry (3) Pictures (7) Videos (1)
Books & Reports (25)
Reports (2) Theses (9) Books (14)
Articles & Preprints (69)
Preprints (37) Notes (1) Drafts (3) Articles (30)

Focus on:

CERN Experiments (6)
ISOLDE (1) ALEPH (5)
CERN Divisions (5)
Theoretical Physics (TH) (4)
Experimental Physics (EP) (1)

ABOUT THIS SITE

Welcome to the demo site of the Invenio, a free document server software coming from CERN. Please feel free to explore all the features of this demo site to the full.

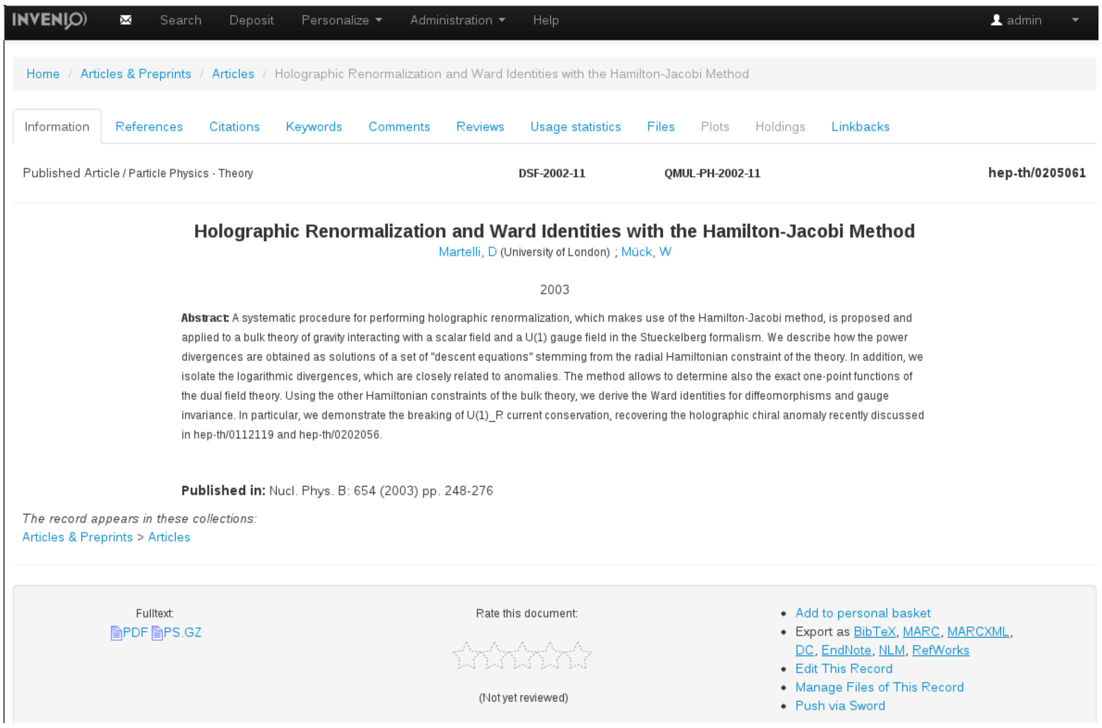
SEE ALSO

Invenio
CERN

Atlantis Institute of Fictive Science :: Search :: Submit :: Personalize :: Help
Powered by Invenio v1.1.1.693-d4158
Maintained by info@invenio-software.org

This site is also available in the following languages:
Afrikaans العربية Български Català Česky Deutsch Ελληνικά English Español Français Hrvatski Galego ქართული Italiano Kinyarwanda Lietuvių Magyar □□□ Norsk/Bokmål Polski Português Română Русский Slovensky Svenska Українська □□(□) □□(□)

Figure 6.2: Screenshot of the main Invenio page in a default vanilla installation



Published Article / Particle Physics - Theory DSF-2002-11 QMUL-PH-2002-11 hep-th/0205061

Holographic Renormalization and Ward Identities with the Hamilton-Jacobi Method
Martelli, D (University of London) ; Mück, W

2003

Abstract: A systematic procedure for performing holographic renormalization, which makes use of the Hamilton-Jacobi method, is proposed and applied to a bulk theory of gravity interacting with a scalar field and a U(1) gauge field in the Stueckelberg formalism. We describe how the power divergences are obtained as solutions of a set of "descent equations" stemming from the radial Hamiltonian constraint of the theory. In addition, we isolate the logarithmic divergences, which are closely related to anomalies. The method allows to determine also the exact one-point functions of the dual field theory. Using the other Hamiltonian constraints of the bulk theory, we derive the Ward identities for diffeomorphisms and gauge invariance. In particular, we demonstrate the breaking of U(1)_R current conservation, recovering the holographic chiral anomaly recently discussed in hep-th/0112119 and hep-th/0202056.

Published in: Nucl. Phys. B: 654 (2003) pp. 248-276

The record appears in these collections:
Articles & Preprints > Articles

Figure 6.3: A record page in an Invenio default installation

Besides storing the metadata of publications, Invenio is capable of storing documents described by the bibliographic records. A separate module called *BibDocFile* is responsible for all manipulations (creating, updating and retrieving) of the stored files. Section 6.3 describes the Invenio document-storage data model in more detail and presents the changes which have been made to allow the storage of figures.

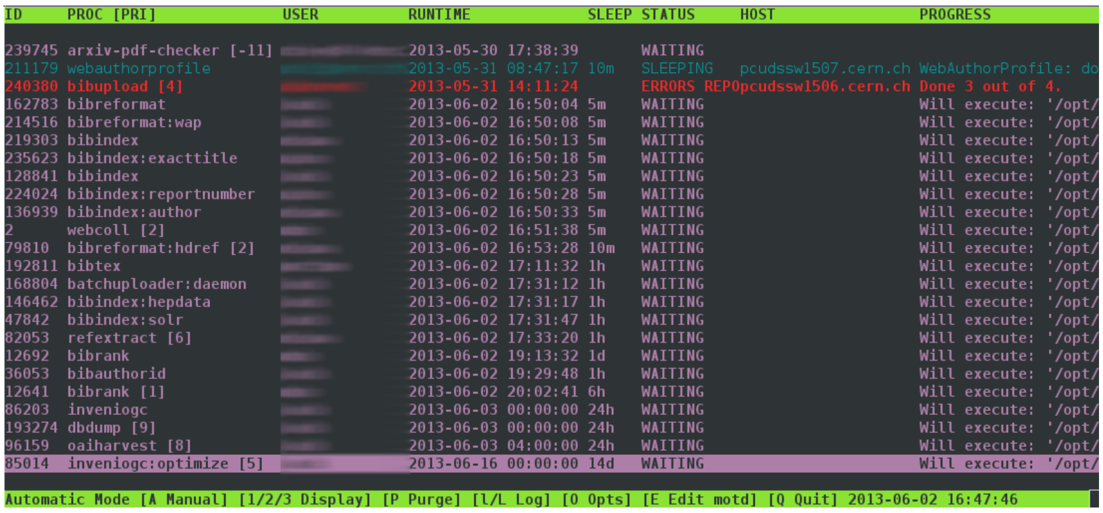
The Acquisition of the (Meta)data

Typically, digital libraries are not static collections of documents which once created, never change. Invenio has been designed and implemented as a system capable of managing very dynamic repositories where new records are added every day, existing records are edited and automatic tasks improve the quality of the metadata. The workflows allowing the ingestion of new publications are closely related to the new workflows allowing ingestion of figures.

In the case of many digital libraries (including INSPIRE), a large fraction of the (meta)data is retrieved from different document repositories. The *BibHarvest* module allows retrieving metadata about publications from different sources implementing the OAI-PMH protocol. *BibHarvest* can be configured to temporarily scan the content of an external repository and to retrieve new relevant records (according to needs either metadata only or both the metadata and the content). The newly-harvested records are transformed (*BibConvert* and *BibFilter* modules) into the MARC XML format and are included in the local collection using the *BibUpload* module. Besides querying external document repositories, Invenio can be configured to allow users to submit their own publications. This functionality is implemented in the *WebSubmit* module.

Including new record in the repository requires modifying the underlying data structures. *BibUpload* is the module which provides abstractions allowing manipulation of bibliographic records. Besides updating the metadata, *BibUpload* is responsible for archiving the metadata of records. It also provides user interface for manipulating the documents attached to records.

The timely execution of tasks, exclusive access to the database and the failover are managed by the *BibSched* module. Modules like *BibUpload* or *BibHarvest* are not executed directly, but instead - added to the queue of *BibSched* tasks. Figure 6.4 shows the interface of BibSched with a number of scheduled tasks, one being suspended and one which has failed.



```
ID      PROC [PRI]                USER           RUNTIME             SLEEP STATUS     HOST                PROGRESS

239745 arxiv-pdf-checker [-11]  [redacted]     2013-05-30 17:38:39       WAITING
211179 webauthorprofile         [redacted]     2013-05-31 08:47:17 10m   SLEEPING   pcudssw1507.cern.ch WebAuthorProfile: do
240380 bibupload [4]            [redacted]     2013-05-31 14:11:24       ERRORS REPOpcudssw1506.cern.ch Done 3 out of 4.
162783 bibreformat              [redacted]     2013-06-02 16:50:04 5m    WAITING                        Will execute: '/opt/
214516 bibreformat:wap          [redacted]     2013-06-02 16:50:08 5m    WAITING                        Will execute: '/opt/
219303 bibindex                 [redacted]     2013-06-02 16:50:13 5m    WAITING                        Will execute: '/opt/
235623 bibindex:exacttitle      [redacted]     2013-06-02 16:50:18 5m    WAITING                        Will execute: '/opt/
128841 bibindex                 [redacted]     2013-06-02 16:50:23 5m    WAITING                        Will execute: '/opt/
224024 bibindex:reportnumber    [redacted]     2013-06-02 16:50:28 5m    WAITING                        Will execute: '/opt/
136939 bibindex:author          [redacted]     2013-06-02 16:50:33 5m    WAITING                        Will execute: '/opt/
2       webcoll [2]             [redacted]     2013-06-02 16:51:38 5m    WAITING                        Will execute: '/opt/
79810   bibreformat:hdref [2]   [redacted]     2013-06-02 16:53:28 10m   WAITING                        Will execute: '/opt/
192811 bibtex                   [redacted]     2013-06-02 17:11:32 1h    WAITING                        Will execute: '/opt/
168804 batchuploader:daemon     [redacted]     2013-06-02 17:31:12 1h    WAITING                        Will execute: '/opt/
146462 bibindex:hepdata         [redacted]     2013-06-02 17:31:17 1h    WAITING                        Will execute: '/opt/
47842   bibindex:solr           [redacted]     2013-06-02 17:31:47 1h    WAITING                        Will execute: '/opt/
82053   refextract [6]          [redacted]     2013-06-02 17:33:20 1h    WAITING                        Will execute: '/opt/
12692   bibrank                 [redacted]     2013-06-02 19:13:32 1d    WAITING                        Will execute: '/opt/
36053   bibauthorid             [redacted]     2013-06-02 19:29:48 1h    WAITING                        Will execute: '/opt/
12641   bibrank [1]             [redacted]     2013-06-02 20:02:41 6h    WAITING                        Will execute: '/opt/
86203   inveniogc               [redacted]     2013-06-03 00:00:00 24h   WAITING                        Will execute: '/opt/
193274 dbdump [9]               [redacted]     2013-06-03 00:00:00 24h   WAITING                        Will execute: '/opt/
96159   oaiharvest [8]          [redacted]     2013-06-03 04:00:00 24h   WAITING                        Will execute: '/opt/
85014   inveniogc:optimize [5]  [redacted]     2013-06-16 00:00:00 14d   WAITING                        Will execute: '/opt/

Automatic Mode [A Manual] [1/2/3 Display] [P Purge] [l/L Log] [O Opts] [E Edit motd] [Q Quit] 2013-06-02 16:47:46
```

Figure 6.4: Screenshot of the BibSched interface

In many cases, records present in the document repository might contain mistakes and need manual curation. This is particularly usual in the case of automatically-generated metadata. *BibEdit*, *MultiEdit* and *BibMerge* modules allow the manual curation of the corpus of stored publications.

Indexing and Assigning to Collections

The aforementioned *WebSearch* module uses internal search indices to retrieve bibliographical records. However, those indices have to be calculated and updated after the collection changes. This task is assigned to the *BibIndex* module. The execution of *BibIndex* is governed by the

BibSched module.

Indices in Invenio are related to a particular metadata field or to a set of them. For example, an installation can define an index which allows searching only inside the MARC field number 100, indicating the name of the main author. Invenio uses forward and reverse indices to search for records. Forward indices map a given search keyword with a set of records which contain this keyword. Reverse indices map a record to the set of keywords. Internally, Invenio uses a bitset data structure to represent search results and entries inside an index. For example, a bitset of records is a data structure storing a single bit for every record present in the repository. If the record is present in the represented set, the bit has value 1. Otherwise, the value is 0. Such a representation allows simple implementation of the set operations on the result set, however creates limitations when dealing with larger collections.

Besides using the search facility, records stored in Invenio can be accessed using the bibliographic collections. Collection is a set of records which share a common property (For example CDS contains collections of books and collection of digital publications for which electronic documents are available). Collections can contain subcollections and thus can be organised in a tree. Navigating through the tree of collections can be used as a method of discovering relevant publications. All the collections have to be defined manually by the administrator of an Invenio installation. In order to specify which records should belong to the collections, an Invenio query has to be provided. The task of recalculating, which records should be placed in a given collection is performed by the BibCol module.

This overview of the structure of Invenio is highly incomplete and many important modules and functionalities and modules have not been mentioned. A much more thorough description can be found on the Invenio website[7].

6.2.2 The Digital Library of HEP

INSPIRE is a project aiming at constructing a digital library of all publications of High Energy Physics (HEP). INSPIRE is a successor of SPIRES [25] database, created since 1970 at the Stanford Linear Accelerator Center (SLAC), providing metadata about preprints. With the advent of WWW, a web interface to the existing resource was created making SLAC the first web site in North America. The introduction of arXiv.org allowed SPIRES to provide not only metadata, but also to display to links to full-text documents. These two services were often perceived as a single system [25]: SPIRES providing a search engine based on the metadata and being an equivalent of a paper-catalogue in libraries; ArXiv.org being a store of articles linked from search results in Spires. The software written in mid 70s was still running at the beginning of the XXI century, becoming increasingly difficult to maintain.

[7] `http://invenio-software.org`

In the era of content-based search engines like Google and user-oriented services, metadata-only based search engines stopped providing the best available service. The INSPIRE project arose as a connection of SPIRES database containing manually curated, high quality records and the Invenio [10] software developed at CERN as a free platform for a digital library repository. INSPIRE, being a service based on a more modern software platform, provides searches based not only on metadata but also on full-text. The development concentrates on the creation of intelligent content-aware tools allowing automatic keywording of records, disambiguation of authors having similar names, or storage and search of figures (the subject of this PhD). INSPIRE will serve a large community of users consisting of the researchers in HEP. The content of INSPIRE comes from various sources. The main corpus of data is being harvested from different digital libraries using the OAI-PMH [17] protocol or obtained directly from publishers. This data is then automatically improved and if necessary, manually curated. ArXiv.org is the single largest source of the publications. INSPIRE also enables its content to be harvested by other digital libraries.

6.3 Data Model for Storing Documents in INSPIRE

Invenio, as many digital library systems, has been built around the concept of a bibliographical record which describes the metadata of a publication. However, the usage of the software suite as a basis of repositories of documents created requirements exceeding the storage of the metadata. Invenio provides functionalities allowing the storage of documents which encode publications (in the terminology of Invenio, those documents are called fulltext documents).
Storing figures and new data preservation efforts emerging in INSPIRE created new challenges related to the storage of custom documents and different types of metadata related to them. Not all of these metadata should be stored in MARC and sometimes, it should describe entities more specific than those described by Invenio records. This section describes the modifications to the internal data model of Invenio, which we have made to enable the storage of figures and other artefacts.

6.3.1 The Previous Storage and Description Model

Before describing the changes to the document storage model of Invenio, we describe the data model existing prior to the modifications. Managing documents is performed by the *BibDocFile* module of Invenio (Described in Section 6.2.1). Figure 6.5 shows the abstract view of the entities related to the storage of documents in Invenio.

Document is the central concept of the data model. It provides the abstraction to store a piece of content in Invenio. The scholarly publication, whose metadata is described by a particular bibliographical record, is the most prominent example of the content stored in

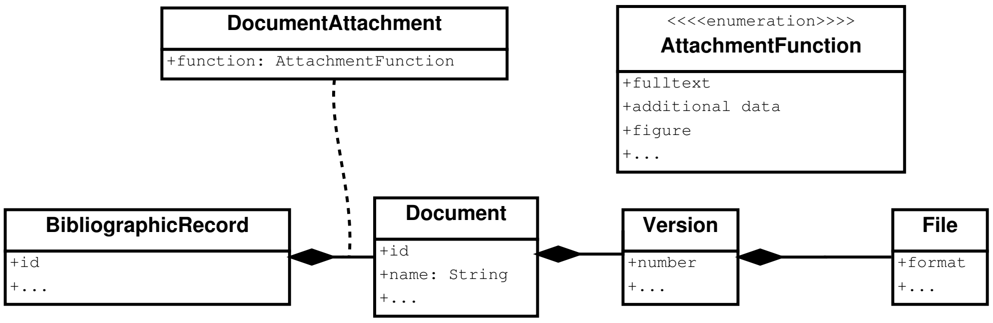


Figure 6.5: The logical view of the old data model of Invenio

Invenio. Prior to the extensions, every document had to be attached to exactly one record and was identified by a name unique in the scope of the attaching record.

The abstract representation of a document ignores the particular encoding of the attached file. Also updating the content of the existing document does not lead to the creation of a new *Document* entity or to overriding the existing one. Instead, it leads to the creation of a new version of the existing document. A document can be understood as a set of versions. Every version is described by a number. The order of those numbers reflects the order of updates of the document. A single document can be encoded in many different ways. For example, a publication can be stored in PDF, PS or ODT files. Each of those files encodes the same content, but differ at the representation level. Each version of a document consists of a set of files encoding the version of the represented content in a particular manner.

At the level of the Invenio API (Application Programming Interface), the concepts of the document storage model were implemented using a number of classes depicted in Figure 6.6. *BibDoc* is a class implementing the concept of a document. At the level of the implementation, versions of a document were not represented by special classes. Instead, *BibDoc* objects were considered as having two dimensions: version and format. These two parameters (a number and a string) had to be provided when manipulating the documents *BibDoc* allowed storage of additional pieces of information that should not be presented in the MARC record of the publication, which is shown to the general audience. This data included different flags used by Invenio software internally and describing the status of the document in the system. The additional metadata was separated from the metadata used by Invenio to manage documents by encapsulating it in an instance of a *BibDocMoreInfo* class. This class implemented accessor methods to different metadata fields. *BibDocMoreInfo* instances were in the one to one relation with the instances of *BibDoc*. An example of the data stored in *MoreInfo* is a flag marking the document as Hidden. Documents marked with this flag were accessible only using the API.

BibDoc instances could be identified by the number of the record to which they are attached

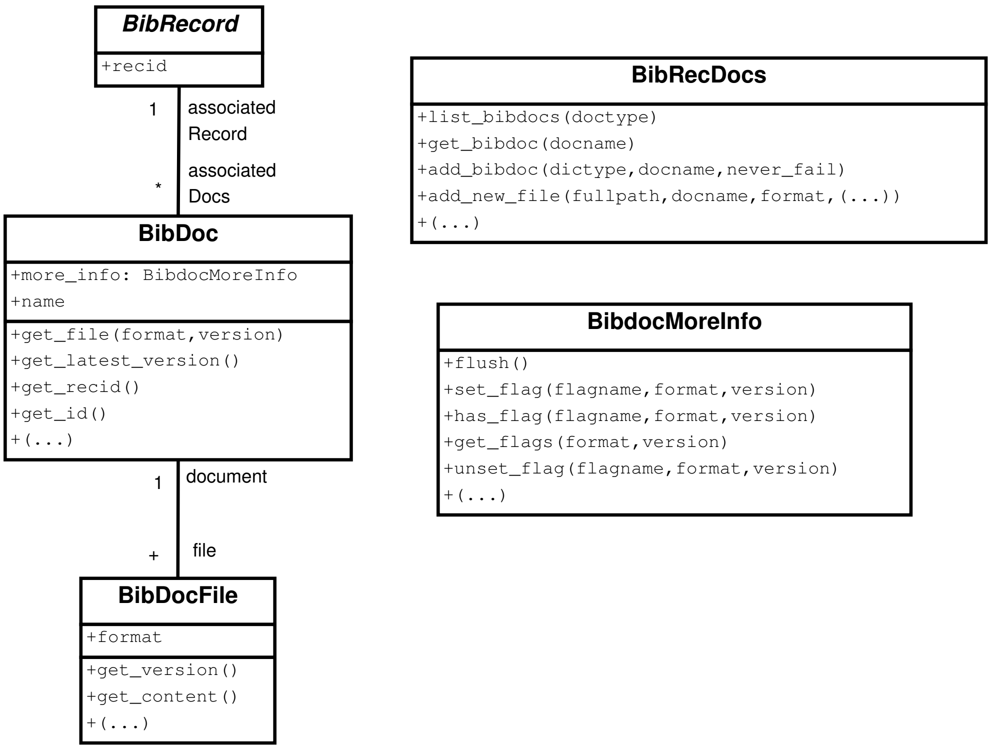

BibRecord
+recid
1
associated Record
associated Docs
*
BibDoc
+more_info: BibdocMoreInfo
+name
+get_file(format,version)
+get_latest_version()
+get_recid()
+get_id()
+(...)
1
document
+
file
BibDocFile
+format
+get_version()
+get_content()
+(...)
BibRecDocs
+list_bibdocs(doctype)
+get_bibdoc(docname)
+add_bibdoc(dictype,docname,never_fail)
+add_new_file(fullpath,docname,format,(...))
+(...)
BibdocMoreInfo
+flush()
+set_flag(flagname,format,version)
+has_flag(flagname,format,version)
+get_flags(format,version)
+unset_flag(flagname,format,version)
+(...)

Figure 6.6: The diagram of classes appearing in the implementation of the old document storage data model

and the name that had to be unique in the scope of this record. Accessing documents attached to a given bibliographical record was possible using a special class called *BibRecDocs*. This class implemented all the operations related to manipulating documents attached to a particular record. Those manipulations included creating new documents, validating document names, suggesting a new unused name, retrieving a document having a given name and so on.

BibDocFile was a class representing the particular representation of a document as a file. Each of its instances encoded a particular format and version. The notion of format was related to the ordinary file format, though it extended it. Besides the main format (like JPEG, PDF, etc.), it was possible to specify a sub-format being an arbitrary string. This mechanism was useful for example in order to store the same graphical file in different resolutions. In this case, a document had more files with the same format but different subformats.

6.3.2 Extension of the Storage Model

The data model previously used by Invenio treated documents only as described content attached to a publication record. However, with the time, the usage of BibDocs started to exceed this scenario. Also the storage of metadata in MARC and in the file system ceased to be sufficient. There exist different types of metadata describing different aspects of documents. Some of these fields are not directly related to documents, but describe the status of the document inside the system. Some metadata is not supposed to be visible to the users of the system. The previous system allowed the storage of these various types of metadata, however in a rather limited fashion. Additionally, the approach of treating documents only as attachments implies that a single document cannot be attached to multiple records; however one bibliographic record can reference many documents.

Figure 6.7 depicts a diagram of concepts appearing in the extended data model. In comparison to the data model presented in Figure 6.5, the main changes include:

- Allowing the establishement of relations between different types of stored objects (The new *Relation* entity)
- Specialisation of the concept of a document to allow custom handling of specific types of objects.
- Attaching objects to possibly many records rather than always exactly one (the relation between a *Record* and a *Document* has been changed from composition to aggregation)
- Storing standalone objects not attached to records.

Most of the above extensions are used to store and to describe the figures.

A number of changes at the API and the implementation level had to be done to implement the new data model. When making changes at the API level, we were trying to minimise the

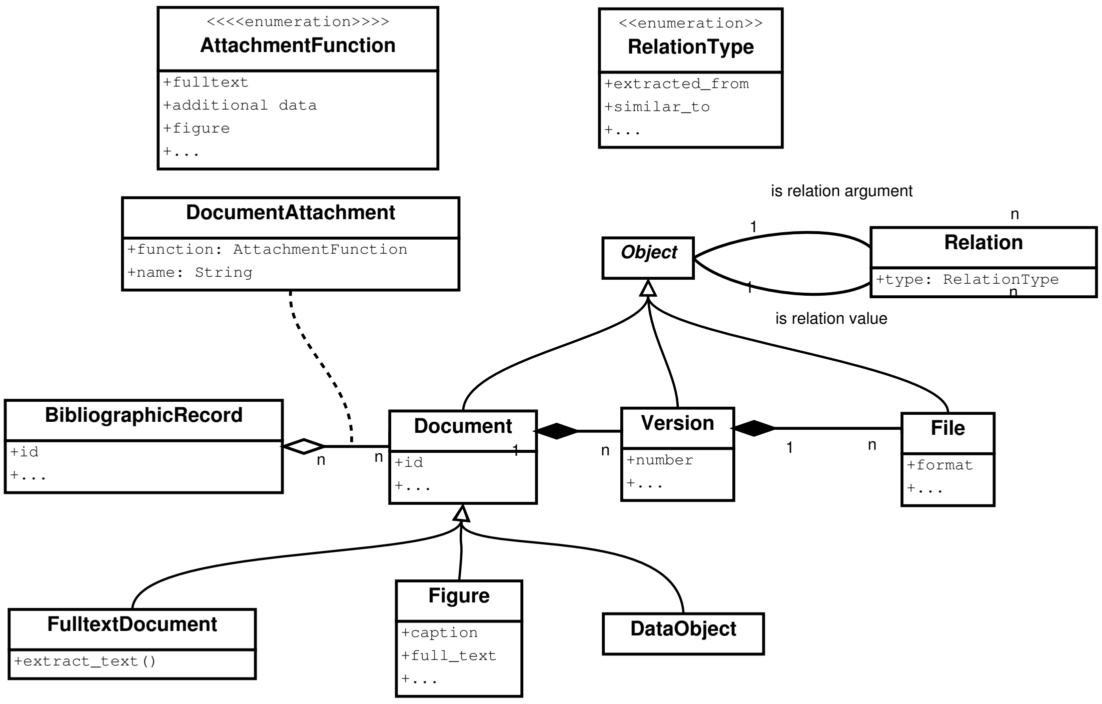


Figure 6.7: Concepts present in the new data model

impact on the existing parts of Invenio which use the document storage. Nonetheless, a number of radical changes were necessary. The implementation features scripts converting documents stored in the old model into the new one. Additionally, there exists a fallback mode that allows reading documents not complying with the new description.

The main change to the *BibDoc* class was to allow the representation of documents independently from the bibliographical records. This meant that a single document can now be attached to multiple records and that multiple records can reference the same document. Such a modification required certain metadata fields to be moved between different classes. The *type* attribute, which described the role of a document in relation to the particular record, was in the old model a part of the document entity. In the new model, this attribute appears as a part of the link between the record and the document. A similar change has been made to the *name* attribute. Many records may refer to the same document using different names. Besides record-unique names, documents are assigned an identifier, which is unique and persistent within the scope of a single Invenio installation.

Previously, the *BibDoc* class was very much oriented towards dealing with documents containing the text of the publications described by bibliographical records. The *BibDoc* class provided for example methods allowing the extraction of the text of publications encoded in

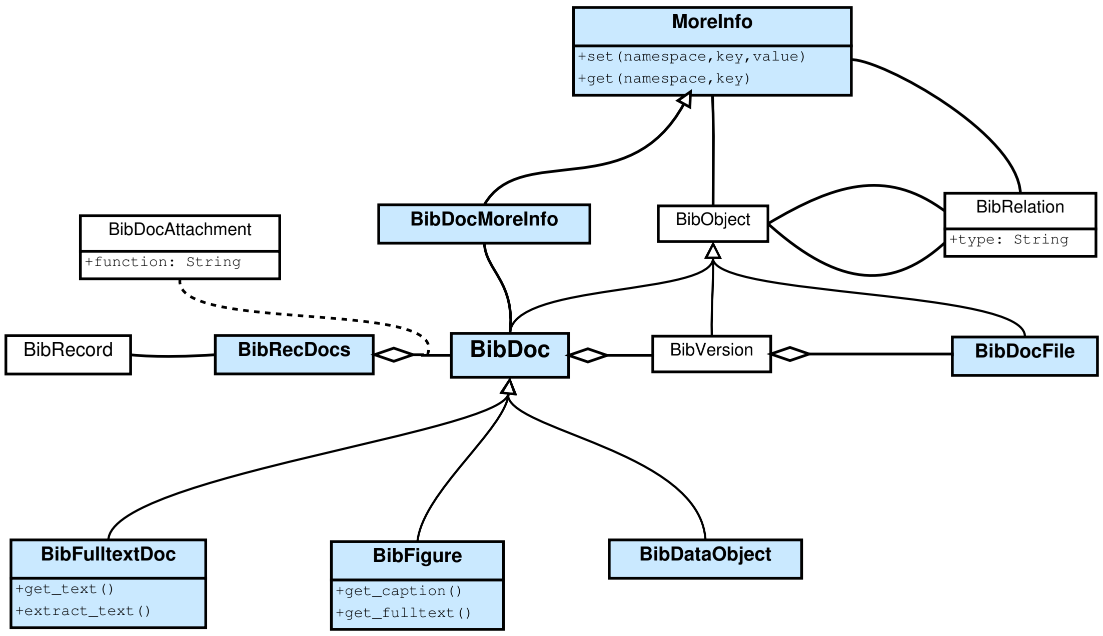


Figure 6.8: Classes implementing the new data model in Invenio

different formats. At the level of the abstraction allowing access to general documents, such a specialisation was not desired. In the new model, a similar specialisation of class behaviours may be achieved by permitting the creation of subclasses of the *BibDoc* class. The creation of appropriate instances of the *BibDoc* class proceeds according to the factory design pattern. The internal Invenio plugin system is capable of detecting the available implementations of a document and to decide which one is the most appropriate in a particular situation. This decision can be done based on an additional type field, indicating the type of a document in a general context rather than within a record. We implemented a number of concrete subclasses which are capable of managing the fulltext documents present in Invenio. A specialisation of the *BibDoc* class is also used to manage graphics. *BibRecDocs* is, similarly to the situation before the introduced changes, an interface that allows obtaining access to the documents related to a given bibliographical record. Instances of this class can be also use to create new documents which will be automatically attached to the given record. This class does not provide functionalities for managing unattached documents.

BibRelation is a new class modelling the concept of generic relations between documents. Relations are established between particular versions of documents. The role of two related documents is not symmetrical. The first document specified in a *BibRelation* instance is called the source of the relation and the second is called target. An instance of the relation contains information about the type, the source and the target *BibDoc* versions. The type of the relation is a string-valued field which describes, how two documents are related. For example, if one document has been created as a result of processing of another document, the *type* could be specified as “extracted from”. Each *BibDoc* instance provides an API to retrieve its incoming and outgoing relations.

One of the largest changes in the data model involved the introduction of the *MoreInfo* class. *MoreInfo* allows the storage of additional non-MARC metadata. In the previous model, the *MoreInfo* instances were closely related to particular instances of *BibDoc*. The types of metadata which could be stored were limited to those predefined in the code of the *BibDocMoreInfo* class. There were specialised functions allowing the storage and modification of certain metadata fields (such as for example access permissions to a document). The new data model made two fundamental changes to the subsystem allowing the storage of the custom metadata. Firstly, *MoreInfo* instances can store arbitrary types of metadata. Every bit of data stored in *MoreInfo* must only satisfy the requirement of being serialisable using the cPickle[8] Python module. Metadata fields are identified using string identifiers. In order to decrease the risk of name collisions (for example in the case when multiple modules try to save under the same name), *MoreInfo* provides the mechanism of name spaces. Secondly, in order to provide a more general metadata storage system, *MoreInfo* instances can now be attached to a wider range of entities, like links between documents, versions or formats. The backwards compatibility with the previous version of the custom metadata storage using dedicated methods for each type of stored metadata has been achieved by providing a specialised *BibDocMoreInfo* class. It implements the previously available API on top of the generalised data structure. Instances of *BibDocMoreInfo* can be retrieved using a specialised accessor method of a *BibDoc* instance.

The interface described in this section does not guarantee that every document will be represented by to at most one in-memory instance of *BibDoc* class at the same time. An Invenio installation might be distributed across multiple computing nodes and multiple Python interpreters could be accessing the document at the same time. When an instance is created, it reflects the state of the object from the moment of instance creation. All the modifications to objects have to be explicitly saved using appropriate flush functions. Invenio solves all the problems of synchronisation by executing data-modifying tasks in a synchronised manner, using the *BibSched* scheduler. In particular, all the modifications to documents and records should be executed using the *BibUpload* module. A more detailed description of the documents-updating interface is presented in Appendix B.

6.3.3 Describing Figures in Invenio

The previous chapters of this thesis described methods of extracting and annotating figures in a general setting. Implementing the extended data model formed the first step towards the

[8]http://docs.python.org/library/pickle.html

management of figures inside an Invenio based repository. However, the data model provides a very general abstraction which can be used for many different purposes. An additional level of abstraction is needed to relate figures with the underlying concepts of Invenio. In this section we describe how the new data model can be used to store figures and their annotations.

After our modifications, Invenio allows the implementation of custom specialisations of the *BibDoc* class. Figure is a particular type of a document and thus, at the API level, it can be represented as a subclass of *BibDoc* called *BibFigure.* This design allows us to reuse the Invenio capabilities of storing documents and of automatically managing their versions as instances of *BibDoc* are stored in the persistent storage of Invenio. Additionally to a document, every figure is described by an Invenio MARC record, which makes figures the first-category objects in the Invenio ecosystem.

The link between the original publication and the figure is also represented in a double manner: first, by a *BibRelation* connecting the document of a figure and the original document of a publication; second, by making a reference inside the MARC field of the figure record. Creating only the MARC link would not be sufficient because we need to maintain connections between particular versions of documents and in Invenio the management of record versions is decoupled from the management of versions of the documents.

The representation by a record together with the attached document tries to abstract from the context in which a figure has been included in a publication. At the same time, we wanted to provide a simple API describing figures in a complete way without unnecessary references to additional entities. Such a state can be achieved by the usage of the dependency injection design pattern. Some of the metadata of a figure are stored with the document; however others are attributes of the relation between the document and the figure. A complete description of a figure in a particular context is provided by a pair consisting of a *BibDoc* describing the figure and a *BibRelation* connecting this document with the one from which a figure has been extracted. At the implementation level a *BibFigure* object can be contextualised with a *BibRelation* object. When manipulating the described figure, all the manipulations dependent on the context of the publication will be executed in the context of a particular document. The *BibRelation* object contextualising a given figure can be changed at any time.

The Storage of the Top-level Annotation

The previous section described how the concept of a figure maps to the entities of the Invenio data model. At this place we describe how different types of the metadata describing figures can be encoded in those entities. In the present section we concentrate on the global annotation of a figure. The next section describes how the structural annotation using HFO can be encoded.

As explained in previous chapters, global annotation of a figure is the annotation which describes the entire figure, not its constituent parts. This type of annotation can be dependent on the context of a publication in which a figure has been included (caption, page location, text references etc. . .) or can be context-independent (the complete text of a figure, the related concepts etc. . .). In both cases the method of encoding the annotation is the same. What varies is the place where the metadata should be stored. The annotation being context dependent can be stored in the *MoreInfo* dictionaries describing a relation between a particular version of the figure document and the publication document. The context free annotation can be part of the *MoreInfo* describing a *BibFigure* instance. All annotations are places under the name space key "figures" and identified by hardcoded key strings which can be interpreted by methods of the *BibFigure* class.

The global annotation can be also divided into two categories using different criteria. The annotating values can be of two different types: literals and lists of literals, references to different vocabularies or lists of those references. The storage of literals and their lists is defined by the specification of the *MoreInfo* dictionary. Whenever there is a need to store a reference to a concept from some vocabulary, the *MoreInfo* entry contains a string value encoding the Unified Resource Identifier (URI) of the referenced concepts.

The Storage of the Structural Annotation

The structural annotation uses the concepts of HFO to encode the internal semantics of the figure. Relations between different of those concepts have a form of graph whose edges are defined by the RDF predicates. This section describes how this graph can be encoded using the internal data model of Invenio.

At the API level, concepts of HFO are represented by Python classes.

BibFigure provides a method of encoding those classes as values which can be stored in *MoreInfo* dictionaries. All the structural annotation is independent from the context of the publication, so the structural annotation is stored only in the *MoreInfo* objects assigned to particular version of a *BibDoc* representing a figure. This *MoreInfo* dictionary contains one entry for every HFO concept. This entry consists of a list of objects representing instances of concepts which appear in the description of the figure. Positions in those lists are used as identifiers of the used instances. They are used whenever one object needs to refer to another one.

The structure of the concept-related classes is strictly defined by the structure of concepts of HFO. The attributes of a given class are defined by the HFO predicates which accept instances of this class as argument. Every such predicate defines an attribute inside the class. Predicates allowing multiple values for a single argument are translated to attributes whose values consist of lists. Single-valued predicates result in single-valued attributes. Atomic values of predicates can consist either of literals, references to external vocabularies or references to other instances stored in the scope of a given figure.

The serialisation procedure translates the content of every instance of an object reflecting an HFO concept into a dictionary. Keys in this dictionary are named after attributes of the serialised object. In the case of literals and references to external vocabularies, values are encoded in the same way as in the case of the external annotation. Whenever the value of an attribute refers to a different HFO concept, the position of this object in the global dictionary is provided. The number provides sufficient information because the name of the attribute defines the type of the value object.

6.4 Extraction of Plots in INSPIRE

In Chapter 3, we described a method for automatically extracting figures encoded in PDF documents. However, INSPIRE has also access to LaTeXsources of a large fraction of the stored documents (Mainly those which are acquired from arXiv.org). In many cases, these LaTeXsources can be used to obtain a set of high quality figures. In this section, we describe the process of extracting figures taking into account LaTeXsource files, and the need for using also different sources. We explain how we combine the results of LaTeXand PDF extraction to obtain high-quality figure descriptions. The last section presents the manual tool allowing the correction of errors in the extraction or annotation of figures using a convenient web interface. Figure 6.9 depicts the overview of the infrastructure allowing the ingestion of figures into the INSPIRE database.

6.4.1 The LaTeXExtractor

The LaTeXfigure extraction tool is based on the most used conventions about including figures in the publications. Typically, when an author wants to include a figure, she uses the *figure* LaTeXenvironment. In most of the cases, graphics which constitute the body of a figure are included using the *includegraphics* command. The text of the caption is enclosed inside the *caption* tag and the identifier (Valid only within the scope of the source file) in the *label* tag. The LaTeXextractor reads the source file and searches for these tags. This allows identifying which files correspond to which captions and identifiers inside the source file. Additionally, the extractor searches for all the places, from which the figure is referenced inside the source file.

In many cases, extracting figures directly from the source description of publications allows achieving a high accuracy of the outcomes. However, there are many reasons making the usage only of the LaTeXextractor a less desirable option.

First of all, publications managed by INSPIRE come from many different origins. Not all of them provide access to the source code of publications. Moreover, not all publications are created using LaTeX. PDF is a de facto standard in the scholarly publishing and accessing a PDF version of a publication is much easier. Even if PDF is not available, it can be easily obtained using one of the standard tools converting between different document formats.

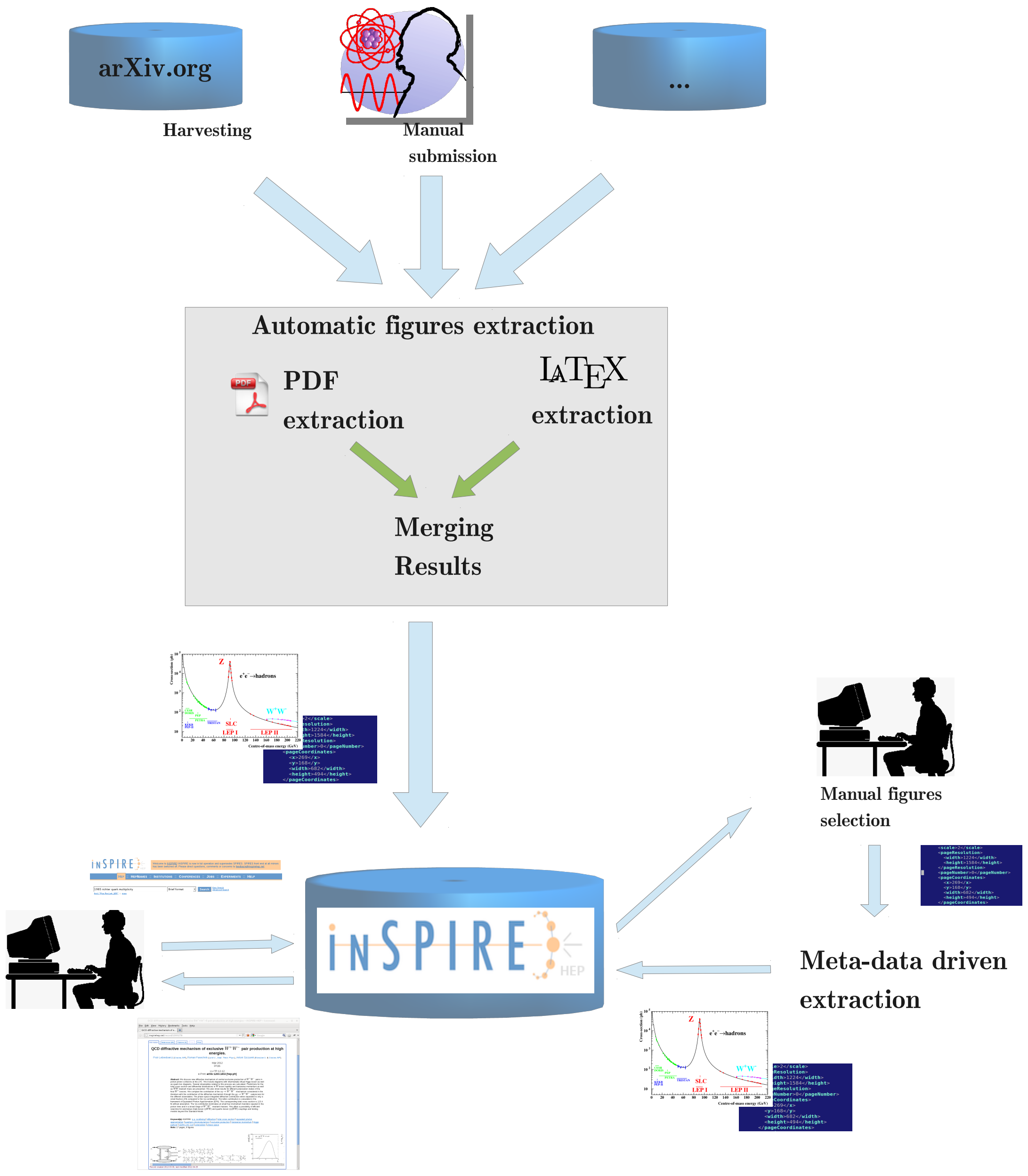


Figure 6.9: The workflow of the ingestion of figures

Even in the case of documents available together with their LaTeXsources, not all the metadata can be extracted from those sources. LaTeXfiles tend to describe the internal structure of the described document rather than provide the display-ready content. The power of the LaTeXsystem comes in part from the fact that many visual-design decisions are made during the compilation. Such properties include the location of figures within a document. Also the numbers of figures are generated during the compilation. Captions and text references can also contain fragments resolved only when generating the final document. These bits of metadata are useful in the final system managing figures and cannot be easily extracted from the source documents.

The method used by the LaTeXextractor allows detecting figures which are included using the most typical method. This approach proves to be successful in the case of a vast majority of publications, but there are numerous exceptions. Typically, figures are embedded in LaTeXdocuments using external graphical files which are included from inside the markup. However, there are a number of LaTeXmodules that allow describing graphics of figures directly from the markup. Such modules are particularly useful when building figures of a more schematic nature (different types of graphs), or when building tables (In this case, LaTeXprovides standard markup). In the area of HEP, such modules allow constructing for example the Feynman Diagrams by describing in a markup language the relations between different parts of a diagram. There is not a central source allowing the access to a list of all possible modules. Constructing a system which would allow the extraction of figures annotated with these modules would lead to a very high complexity and as such, would require much more resources.

Also figures which are included in a publication by the means of including an external graphical file can lead to problems. In some cases, the LaTeXmarkup is used to combine many external files into a single figure. The LaTeXextractor uses a rather simplistic method which makes all of these files appear as separate figures that share the same caption. The result of such an extraction is not always correct. There are also many parameters which can be specified when including an external file. Some of them can considerably modify the display of the file. Maintaining a comprehensive list of markup which modifies the display poses similar challenges as maintaining a list of all possible external modules. Additionally, the commands surrounding the inclusion of graphics can modify the display in an arbitrary manner.

The files referenced from the content of a LaTeXdocument can be encoded in many different formats. PDF documents contain the same encoding method for all included objects. Processing the uniform description is much easier than managing the conversion between formats. Finally, even if the inclusion of figures tends to be standard in the majority of documents, LaTeXdoes not enforce any particular method of achieving this. This increases the likelihood of a failure.

All the aforementioned reasons inspired the decision of using multiple extractors simultaneously. The constructed prototype includes only two extractors. However, in the case of need,

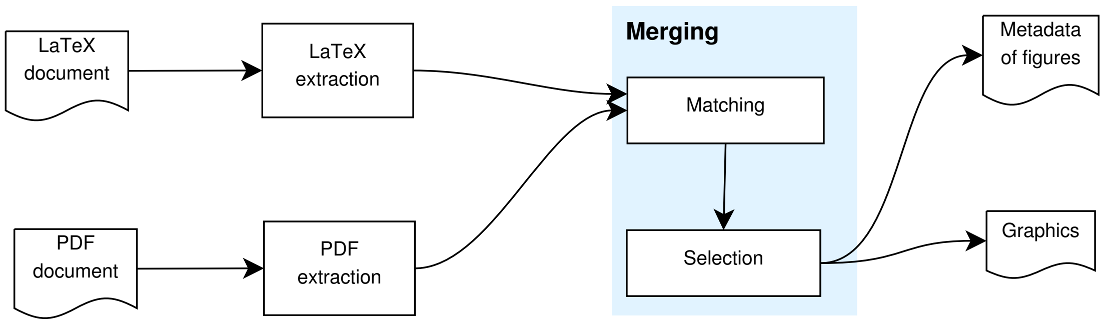


Figure 6.10: The overview of the process of merging figures coming from PDF and LaTeXextractors

this list can be extended.

6.4.2 Merging the Results from the LaTeXand PDF Extractors

This section describes the method of merging outcomes of PDF and LaTeXextractors. The presented method has been implemented in order to combine results coming from the LaTeXand PDF extractors implemented for the usage in INSPIRE. However, presented methods are general enough to be used with figures obtained also from other sources.

Figure 6.10 depicts an overview of the process of extracting figures in INSPIRE. As aforementioned, publications stored in the database of INSPIRE come from different sources differing in characteristics. These sources can provide publications encoding in different formats. Some metadata sources provide documents which can be processed by either of the extractors. The process summarised in Figure 6.10 covers the most complicated case, when both PDF and LaTeXdocuments are available. In other cases, the merging process is omitted and the extracted metadata is automatically uploaded.

At this point, we concentrate on the details relevant to the extraction and processing of figures, omitting the details of the harvesting. During the process of ingestion, the documents together with the describing metadata are first retrieved from the source. Every document is subsequently passed through a figure extractor corresponding to the encoding format. Extracted figures are passed to the merger, which produces the final representation ready for the upload to INSPIRE.

Matching Figures

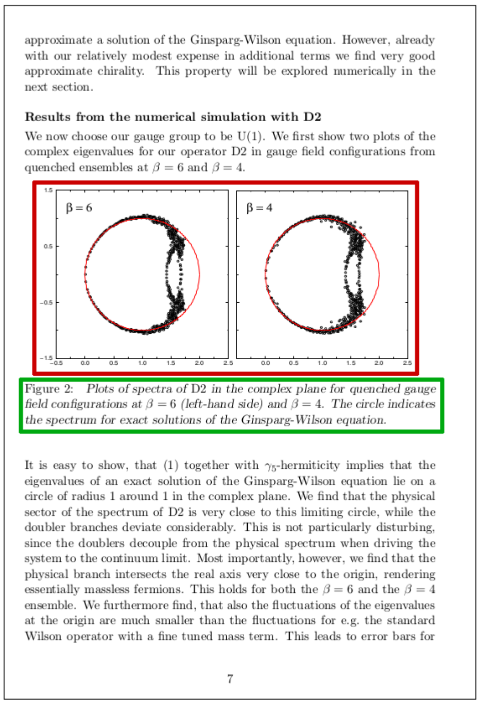
Typically, outputs of different extractors describe the same set of figures. Matching figures from different output sets allows establishing correspondence between figures detected in LaTeXfiles and those extracted from PDFs. The correspondence does not have to be bijective. It might happen that a single figure detected in the results of an extractor corresponds to many figures detected by the other. In such cases, the multiple figures tend to describe subfigures of a larger one. Sometimes, due to errors in the extraction, results from different extractors contain completely different sets of figures. The process of matching the results allows detecting which figures are represented in both result sets and which are missing in one of them.

Figure 6.11 shows hypothetical results of extraction which uses two different methods. In the example, the figure appearing on the first page has been detected by both extractors. However, the PDF extractor detected it as a single entity and the LaTeXextractor detected both plots separately, assigning them the same caption. The task of the matching process is to detect that a single entity from the output of the PDF extractor corresponds to two entities in the LaTeXextractor output. The second page from the example contains a figure which has been correctly extracted from the LaTeXfile, but it does not appear in the output of the PDF extractor. In this case, the merging process should detect that a figure is missing in the PDF output. It is worth noting that the output of the PDF extractor allows us to locate figures inside the publication page (and generate image like in the Figure). However, the LaTeXextractor shows only the file which contains the image and the text of the caption. The location of the detection figure inside the output document (The coordinates of the red and green rectangles in the right part of the Figure) remains unknown.

Establishing relations between figures requires having means of comparing them. The comparison can be done either at the level of metadata or at the level of the image encoding a figure (or using a combination of these). In both cases, the extractor cannot assume that, even in the case of correct extraction from all sources, the representations will be identical. In the case of images, the differences might come from having different rendering engines or using different settings (like anti-aliasing). Also the size and the orientation of the output images might be different.

In the case of the metadata, differences can result from different encoding of the same characters (e.g. LaTeXsource and corresponding Unicode characters like "α" vs "\alpha"). Also some parts of the content (like names of the figures inside the publications e.g. "Fig. 1") cannot be resolved in the metadata coming from LaTeXextractor. In the case of extracting non-linear text blocks (e.g. fragments of mathematical expressions), the order of fragments might vary between extractors.

The above reasons enforced an implementation based on approximate comparisons. For the purpose of the prototype system, we decided to implement a comparison based on a metadata field. There were two main candidates for the metadata field to be used when matching figures: the location and the caption. Matching based on the location intuitively had a higher potential of yielding correct results. However, only the PDF extractor provides the location. Determining the location based on the image of a figure would be difficult. The implementation would have to consider possible transformations of the original image before being included in the output file. Matching using captions was a more risky choice, but easier to implement. The experiments



approximate a solution of the Ginsparg-Wilson equation. However, already with our relatively modest expense in additional terms we find very good approximate chirality. This property will be explored numerically in the next section.

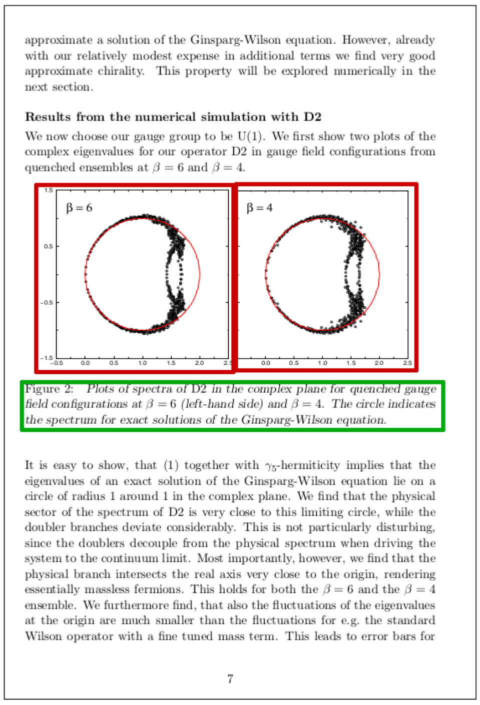
Results from the numerical simulation with D2

We now choose our gauge group to be U(1). We first show two plots of the complex eigenvalues for our operator D2 in gauge field configurations from quenched ensembles at $\beta = 6$ and $\beta = 4$.

Figure 2: Plots of spectra of D2 in the complex plane for quenched gauge field configurations at $\beta = 6$ (left-hand side) and $\beta = 4$. The circle indicates the spectrum for exact solutions of the Ginsparg-Wilson equation.

It is easy to show, that (1) together with γ_5-hermiticity implies that the eigenvalues of an exact solution of the Ginsparg-Wilson equation lie on a circle of radius 1 around 1 in the complex plane. We find that the physical sector of the spectrum of D2 is very close to this limiting circle, while the doubler branches deviate considerably. This is not particularly disturbing, since the doublers decouple from the physical spectrum when driving the system to the continuum limit. Most importantly, however, we find that the physical branch intersects the real axis very close to the origin, rendering essentially massless fermions. This holds for both the $\beta = 6$ and the $\beta = 4$ ensemble. We furthermore find, that also the fluctuations of the eigenvalues at the origin are much smaller than the fluctuations for e.g. the standard Wilson operator with a fine tuned mass term. This leads to error bars for

7



approximate a solution of the Ginsparg-Wilson equation. However, already with our relatively modest expense in additional terms we find very good approximate chirality. This property will be explored numerically in the next section.

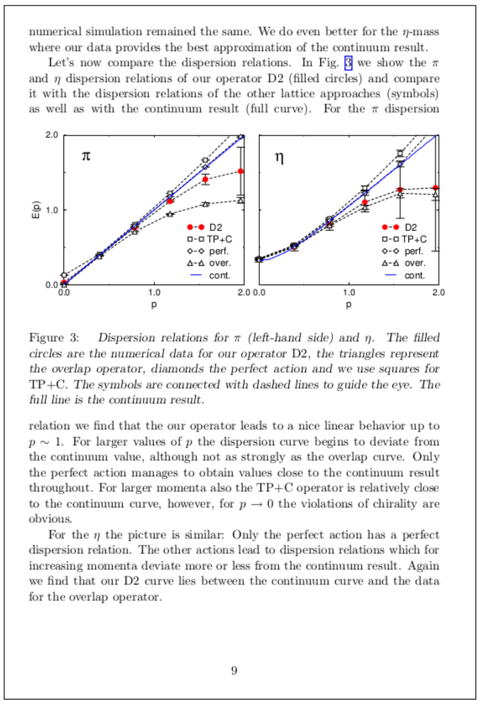
Results from the numerical simulation with D2

We now choose our gauge group to be U(1). We first show two plots of the complex eigenvalues for our operator D2 in gauge field configurations from quenched ensembles at $\beta = 6$ and $\beta = 4$.

Figure 2: Plots of spectra of D2 in the complex plane for quenched gauge field configurations at $\beta = 6$ (left-hand side) and $\beta = 4$. The circle indicates the spectrum for exact solutions of the Ginsparg-Wilson equation.

It is easy to show, that (1) together with γ_5-hermiticity implies that the eigenvalues of an exact solution of the Ginsparg-Wilson equation lie on a circle of radius 1 around 1 in the complex plane. We find that the physical sector of the spectrum of D2 is very close to this limiting circle, while the doubler branches deviate considerably. This is not particularly disturbing, since the doublers decouple from the physical spectrum when driving the system to the continuum limit. Most importantly, however, we find that the physical branch intersects the real axis very close to the origin, rendering essentially massless fermions. This holds for both the $\beta = 6$ and the $\beta = 4$ ensemble. We furthermore find, that also the fluctuations of the eigenvalues at the origin are much smaller than the fluctuations for e.g. the standard Wilson operator with a fine tuned mass term. This leads to error bars for

7



numerical simulation remained the same. We do even better for the η-mass where our data provides the best approximation of the continuum result.

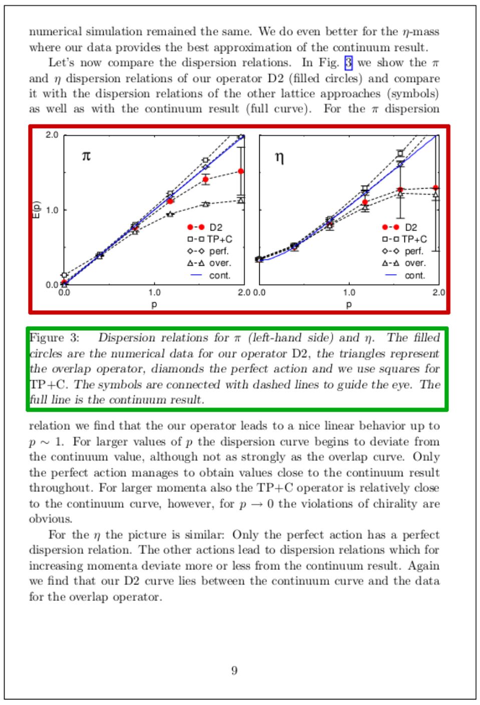
Let's now compare the dispersion relations. In Fig. 3 we show the π and η dispersion relations of our operator D2 (filled circles) and compare it with the dispersion relations of the other lattice approaches (symbols) as well as with the continuum result (full curve). For the π dispersion

Figure 3: Dispersion relations for π (left-hand side) and η. The filled circles are the numerical data for our operator D2, the triangles represent the overlap operator, diamonds the perfect action and we use squares for TP+C. The symbols are connected with dashed lines to guide the eye. The full line is the continuum result.

relation we find that the our operator leads to a nice linear behavior up to $p \sim 1$. For larger values of p the dispersion curve begins to deviate from the continuum value, although not as strongly as the overlap curve. Only the perfect action manages to obtain values close to the continuum result throughout. For larger momenta also the TP+C operator is relatively close to the continuum curve, however, for $p \to 0$ the violations of chirality are obvious.

For the η the picture is similar: Only the perfect action has a perfect dispersion relation. The other actions lead to dispersion relations which for increasing momenta deviate more or less from the continuum result. Again we find that our D2 curve lies between the continuum curve and the data for the overlap operator.

9



numerical simulation remained the same. We do even better for the η-mass where our data provides the best approximation of the continuum result.

Let's now compare the dispersion relations. In Fig. 3 we show the π and η dispersion relations of our operator D2 (filled circles) and compare it with the dispersion relations of the other lattice approaches (symbols) as well as with the continuum result (full curve). For the π dispersion

Figure 3: Dispersion relations for π (left-hand side) and η. The filled circles are the numerical data for our operator D2, the triangles represent the overlap operator, diamonds the perfect action and we use squares for TP+C. The symbols are connected with dashed lines to guide the eye. The full line is the continuum result.

relation we find that the our operator leads to a nice linear behavior up to $p \sim 1$. For larger values of p the dispersion curve begins to deviate from the continuum value, although not as strongly as the overlap curve. Only the perfect action manages to obtain values close to the continuum result throughout. For larger momenta also the TP+C operator is relatively close to the continuum curve, however, for $p \to 0$ the violations of chirality are obvious.

For the η the picture is similar: Only the perfect action has a perfect dispersion relation. The other actions lead to dispersion relations which for increasing momenta deviate more or less from the continuum result. Again we find that our D2 curve lies between the continuum curve and the data for the overlap operator.

9

Figure 6.11: Hypothetical results of outcomes from the PDF and LaTeXextractors. Figures are marked with red frames and captions by green ones.

with automatic extraction from both PDF and from LaTeXshowed that the extraction of captions from the both sources can be conducted with a high reliability.

As mentioned earlier, captions extracted by the PDF figures extractor and by the LaTeXextractor might vary in small details. Before being compared, caption strings are preprocessed in a manner which is expected to minimise the differences between captions of the same figure. The identifiers appearing at the beginning of captions are first removed as LaTeXsource cannot contain them in the same form as the PDF outcome. Differences of the captions resulting from different encoding of non-standard characters are minimised by removing all the characters outside the standard ASCII range. Also all the LaTeXcommands are removed. Additionally, strings are stripped from all the white characters. Caption strings preprocessed in this manner cease to be human-readable and loose important parts of the encoded information. However, in most cases they are sufficient to be matched between each other and show much higher similarity.

In some cases even the preprocessing can keep the corresponding strings different. The experiments have shown that using the exact matching technique was not an optimal solution for matching figures. Instead, the module uses a similarity measure which tells how similar two strings are.

The most popular similarity measure used to compare strings is the edit distance, also called the Levensthein distance [13]. After retrieving all the figure captions and preprocessing them, the merging algorithm calculates the edit distance between every pair containing a caption from PDF and a caption from LaTeX. If the distance is small enough, captions are considered to describe the same figure.

At every stage of the execution, the matching algorithm maintains an undirected graph, which models the relations between figures. Initially, every figure detected by either of the figure extractors forms a separate vertex and there are no edges in the graph. Whenever two figures are considered to be the same, an edge is created between their corresponding vertices. At the end of the execution, the graph describes all the relations between the extracted figures. Every connected component of this graph describes a single figure appearing at the output of the merger.

Merging the Metadata

Every vertex of the constructed relations graph is annotated with a full figure metadata. However, only one output entry is produced for every connected component of the final graph. It is not obvious, how should the descriptions of separate vertices be merged into a single one.

The present version of the merging module exploits the a priori assumption that the quality of different bits of metadata depend on the method of the extraction. For example, captions extracted from the PDF file are more likely to contain text in the same form as appearing inside the publication than those from LaTeXsource. PDF provides coordinates of a figure within the publication document. The same metadata is not available in the results from PDF. On the

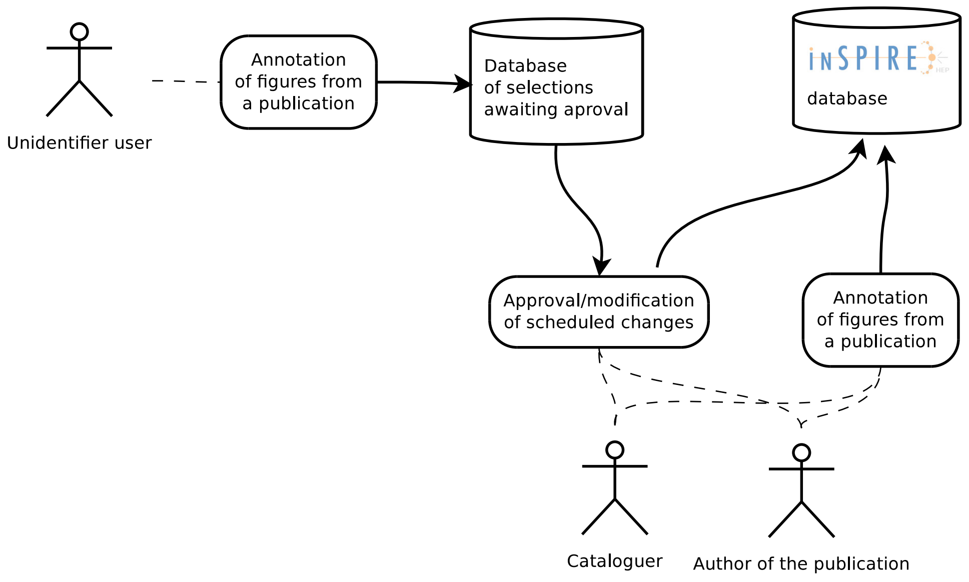

Unidentifier user
Annotation of figures from a publication
Database of selections awaiting aproval
INSPIRE database
Approval/modification of scheduled changes
Annotation of figures from a publication
Cataloguer
Author of the publication

Figure 6.12: The workflow of manual editing of figures

other hand, if the LaTeXextractor provides text references, they are considered to be better than those from PDF. The merger always chooses the best source of the metadata and includes the information in the output.

6.4.3 The Manual Extractor

In the previous sections, we described the method of automatically retrieving figures from various sources. Automatic procedures, however efficient, in some cases can provide incorrect results. INSPIRE in a big part relays on the manual curation of the metadata. Also a tool for manual selection and annotation of figures has been included in Invenio and should be integrated in the document-processing workflows of INSPIRE. Figure 6.9 shows the role of the manual extractor in the process of including figures in the INSPIRE database. Figure 6.12 focuses on the flow of control when different types of users interact with the manual tool.

There are three main types of users who can access the extraction interface. Cataloguers are the INSPIRE staff (Either CERN or other involved laboratory), whose main task is to improve the annotation of bibliographic records. Authors are the users who have been identified as authors of a particular publication. INSPIRE can discover that the user is author of publication for example by using the integration with arXiv.org, which is the source of much of the INSPIRE content. The third group of users consists of regular INSPIRE users. They are not identified in any way. Actions performed by Cataloguers and Authors can be trusted, whilst metadata coming from the regular users should be additionally verified before being included in INSPIRE.

In the most typical scenario, an unauthenticated user interacts with the manual annotation after seeing a page describing a publication and discovers that figures are not presented correctly. We cannot assume that the input provided by a typical user is correct and that there was no malevolent intention behind the editing. Thus, results of such modifications should not be automatically uploaded into the main database but rather enqueued for approval by the cataloguer or (if available), by the author. If the identity of the author is known, she can be notified that someone has suggested modifications to her publication and should be allowed to verify the correctness of this data.

Authors can access the interface in the same manner. Creators of a publication are likely to check the manner in which their work is displayed in various information systems and they tend to have a high motivation for improving the mistakes. If INSPIRE is able to identify the editing user as the author, it can trust the updated annotation and upload it directly to the central database. However, authors can be unaware of certain metadata annotation standards of INSPIRE (not all of them can be enforced by the interface) and thus there exists a chance that they will make mistakes. After a figure has been edited by an author, a notification should be sent to cataloguers. Provided there is a sufficient time, cataloguers can verify the metadata (already uploaded to INSPIRE and visible to the users).

Cataloguers have a completely different motivation for editing the annotation of records. They are specialised in creating annotation complying with all the INSPIRE standards. Cataloguers can access the annotation interface in order to respond to data correction requests, in order to verify the correctness of the modifications performed by other users or as a routine checking across the newly harvested collections. Changes made by cataloguers are applied directly to the central database.

As the manual extraction and annotation tool has to be available to unauthenticated users, we decided not to allow users to upload their own files as figures. One of the main assumptions about the extraction of figures present in the publications was that we want to give access to figures which are present in the exact publication file which is managed by INSPIRE. The web-based interface presents a view of the entire publication document. User always works on a single page, which is displayed in a larger window. Miniatures of the other pages are presented on the side, providing a preview of the content and allowing users to easily change the current page. In order to select a figure, user needs to draw a rectangle around the area, where the figure is located inside the paper. A similar mechanism is used to select the boundary of the caption. After all the selections are approved by the user, the system automatically generates all the images. This is done using the original document of the publication. This approach assures that only the content of the original publication can be included in the repository. It also makes the extraction simpler for the end users as they do not have to use any external Ellis to prepare the image files. Additionally, by performing all the file-generation inside INSPIRE, we assure that the stored files will be encoded in uniform manner.

Overview of the Main Functionalities

Figure 6.13 shows the interface that allows selecting figures from a sample publication. The central part of the view is occupied by a view of the publication page. The panel on the left shows all pages of the publication. The top panel displays already existent figures. The right side of the interface is used to edit the properties of the figure.

The page view is a control giving access to a publication page. The user is given the possibility of zooming the page and scrolling it in order to make the interesting part of it visible. It is also possible to rotate the page by an arbitrary angle. This feature is particularly useful when dealing with publications which have been created by scanning their paper versions. In such cases, the image of the page is sometimes rotated by a small angle due to imperfections in the scanning process. Also some figures are rotated by the angle of 90 degrees with respect to the natural page position. Allowing rotation of the publication page makes it possible to see and to select these figures in their natural position.

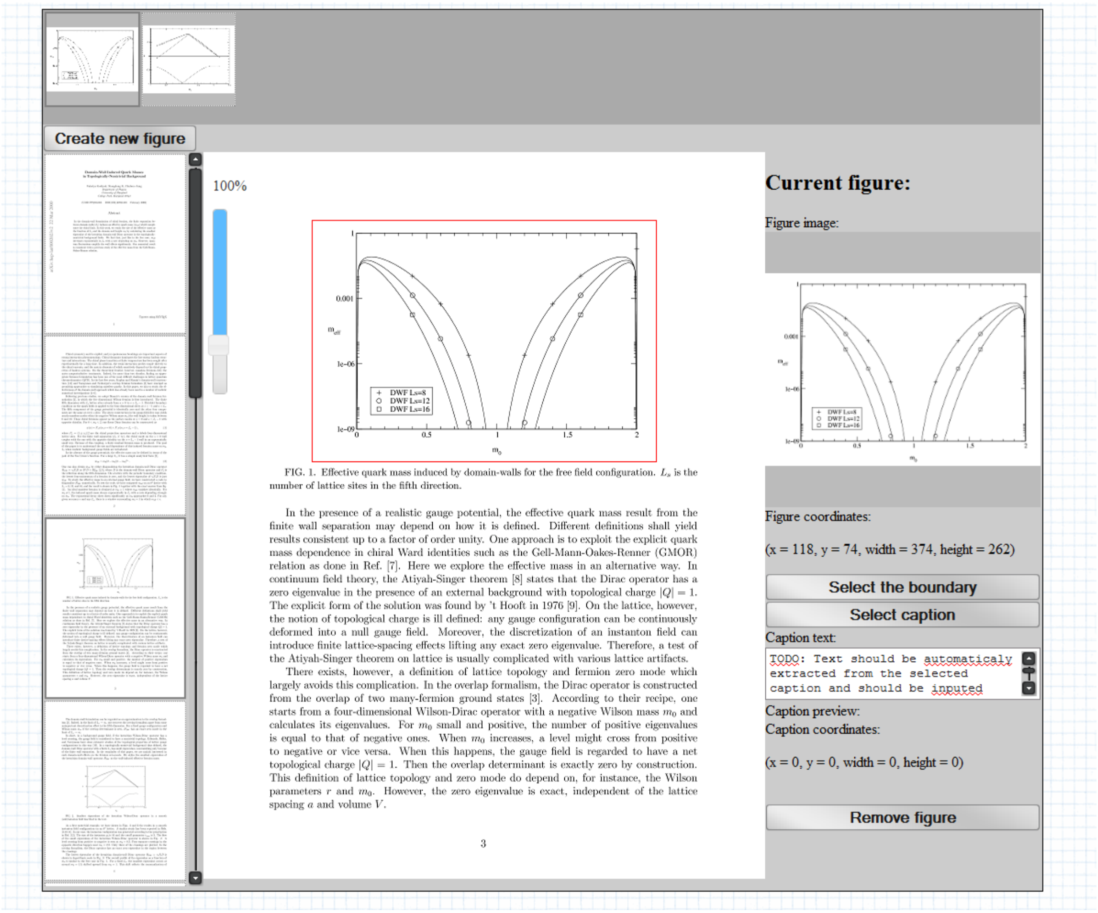
The manual figures extractor uses the notion of the current figure. Current figure is the one which is being edited. It is presented in the page view interface in a special way and the figure metadata panel displays properties of this figure. Users can change the current figure by choosing another one in the figures panel or by creating a new figure. If users click on a miniature of the figure, the page view switches to the page, where the figure is located. Creating new figure empties the metadata fields in the figure properties panel and creates an empty box inside the figures panel.

The panel with figure details allows modifying the parameters of a figure. At present, the tool allows annotating figures only with the most important features, but future extensions could allow integration of more elements of the semantic annotation. At present, users are given an option to select the boundary of the figure. After choosing to select the location of a figure, user can draw a rectangle inside the page view area. A miniature of the selected figure will be displayed inside the metadata panel. Selection of the caption proceeds in the same manner. However, after the user indicates the location of the caption, the manual extractor attempts to extract the text present inside the publication. The text is inserted into a text box, which can be modified.

The last element of the metadata panel is the button to submit the changes which have been made.

The Architecture

On the technical side, the manual figures extraction works inside a web browser. The interaction with the server is provided by means of AJAX requests and is limited to sending the metadata.



```
TODO: Text should be automaticaly
extracted from the selected
caption and should be inputed
```

FIG. 1. Effective quark mass induced by domain-walls for the free field configuration. L_s is the number of lattice sites in the fifth direction.

In the presence of a realistic gauge potential, the effective quark mass result from the finite wall separation may depend on how it is defined. Different definitions shall yield results consistent up to a factor of order unity. One approach is to exploit the explicit quark mass dependence in chiral Ward identities such as the Gell-Mann-Oakes-Renner (GMOR) relation as done in Ref. [7]. Here we explore the effective mass in an alternative way. In continuum field theory, the Atiyah-Singer theorem [8] states that the Dirac operator has a zero eigenvalue in the presence of an external background with topological charge $|Q| = 1$. The explicit form of the solution was found by 't Hooft in 1976 [9]. On the lattice, however, the notion of topological charge is ill defined: any gauge configuration can be continuously deformed into a null gauge field. Moreover, the discretization of an instanton field can introduce finite lattice-spacing effects lifting any exact zero eigenvalue. Therefore, a test of the Atiyah-Singer theorem on lattice is usually complicated with various lattice artifacts.

There exists, however, a definition of lattice topology and fermion zero mode which largely avoids this complication. In the overlap formalism, the Dirac operator is constructed from the overlap of two many-fermion ground states [3]. According to their recipe, one starts from a four-dimensional Wilson-Dirac operator with a negative Wilson mass m_0 and calculates its eigenvalues. For m_0 small and positive, the number of positive eigenvalues is equal to that of negative ones. When m_0 increases, a level might cross from positive to negative or vice versa. When this happens, the gauge field is regarded to have a net topological charge $|Q| = 1$. Then the overlap determinant is exactly zero by construction. This definition of lattice topology and zero mode do depend on, for instance, the Wilson parameters r and m_0. However, the zero eigenvalue is exact, independent of the lattice spacing a and volume V.

3

Figure 6.13: The interface of the manual figures selection tool.

Most of the logic is implemented on the client side using Javascript. All the graphical operations (rotations, cropping of the images etc...) are performed using Cascade Style Sheets (CSS), which is dynamically modified using Javascript.

Metadata-driven Extraction

The results of the manual extractor consist only of the position of a figure together with its basic metadata. However, INSPIRE saves the images of figures. Saving results of the manual extraction in INSPIRE requires an additional intermediate step of producing images based on the annotation. This task is performed by a modified version of the PDF extractor, which accepts the coordinates of a figure and returns the image in all necessary formats. Before uploading to the INSPIRE database, the results of the metadata-driven extractor can be processed by the module extracting semantics of figures.

The Usage with a Large Corpus of Publications

INSPIRE contains a number of bibliographical resources which do not satisfy the prerequisites of the automatic extraction process. Such resources include collections of older publications which has been scanned from their paper versions and included in the collection of INSPIRE. The manual extraction tool can be used to process them. However, at present INSPIRE does not have a sufficient amount of spare working time of the cataloguers, who could process, especially larger, collections.

Recent years have witnessed the development of services that allow allocating on demand the necessary human power [32]. An additional use case of the manual extraction tool could involve using such a service to process larger collections of data. This scenario might seem similar to the case of having unauthenticated users select figures from INSPIRE publications. However, there are a number of characteristics which make this use case substantially different.

In the case of Mechanical Turk or a similar system, the work needed to be finished has to be provided by the system rather than selected by the users themselves. In a typical usage inside INSPIRE, the user sees a publication (typically, related to her area of expertise) and decides that there is need of improving the data related to this publication. Using a cloud-like service would enforce INSPIRE to implement a module dividing the dataset into smaller sets which could be automatically dispatched to separate workers. Additional attention should be devoted into the granularity of a single extraction task. INSPIRE could split publications into separate pages and present them to workers or to provide them the entire papers. The first approach avoids difficulties with larger publications (In some cases, a single publication can have many hundred pages). The second approach has a potential of resulting in more uniform results. Figures inside a single publication tend to be similar (or at least included in a similar manner) and presenting the entire publication to a worker would result in having a more uniform annotation.

The selection of figures is not an overwhelmingly difficult task, nonetheless, workers using the system are much more likely to make mistakes than an average INSPIRE user. Every worker should be provided with a good tutorial on how to use the tool and how to correctly annotate the data. This could involve usage of preannotated sample publications, and asking workers to annotate them, to evaluate the ability of a particular worker to process the data.

In a typical INSPIRE scenario, annotation results are not automatically included in the main database, but rather enqueued for a manual revision by an authorised user. Large scale crowdsourcing has the potential of generating revisions queues which would exceed the capacity of the cataloguers. Additional development should be devoted to some type of automatic curation system. The same sample could be for example dispatched to multiple workers and results could be compared. In the case of large differences, the same sample could be presented to an additional worker and then results compared with those already obtained. If results obtained from different workers were similar, the annotation could be considered correct.

Additionally, remembering the identity of a worker could allow assigning every worker with a number signifying her reputation. If the annotation generated by a particular worker has been approved, the reputation could grow. Similarly, if the annotation has been rejected, the reputation could be decreased. The measure of reputation could be used when estimating the quality of new submissions by the same worker.

6.4.4 What Happens When a Publication is Updated

Harvesting publication from OAI sources (such as arXiv.org) does not only create new entries in the database. Sometimes existing publications have to be updated, causing the existing metadata to be partially or completely updated. In the case of records, INSPIRE features a complete workflow that enables the automatic management of such situations. If the algorithm cannot decide the correct action to perform, a cataloguer is asked to provide manual input. Updating records in Invenio does not cause the old version to disappear from the database. Instead, a new version is created and the old one is archived.

Updates can also affect figures. The new version of a publication file can contain different figures with respect to the previous one. Also the same figures can appear in newer versions or the metadata of the same figure can be changed. It is important to correctly manage the scenarios of an update.

After a new version of a publication is retrieved from an external source, it goes through the entire extracting and processing mechanism. This includes the extraction of figures. The new available figures are compared with the set of figures already present in INSPIRE. For every such figure, a new version of the describing record is created (together with the metadata description). Also records for new figures are created.

Records of figures which have been removed in the new version of a publication remain intact, only the link with the main record is removed. In the future, the view of figures attached to a bibliographical publication should be updated, so that figures that no longer belong to a publication could be visible.

6.5 Integration with the Search System of Invenio

Encoding figures as bibliographical records enables the usage of the internal Invenio search engine for the figure retrieval. At the time of writing, Invenio does not provide a semantic storage, which would allow searching using the full semantic annotation as described in previous chapters. Additionally, a fully semantics-aware search would be much more resource-consuming than the existing solution of INSPIRE.

In order to avoid too deep modifications at the heart of the Invenio software, we have decided to use a more hybrid approach of indexing elements of the precalculated semantic annotation using Invenio indices. This allows the system to take advantage of the improvements of accuracy characteristic for the semantic search, when using selected metadata queries. However, such a solution does not have the full flexibility of a triplestore search engine as it limits the usage to the predefined queries.

This section describes this hybrid approach together with a fully flexible scenario of using a separate semantic search engine and merging its results with those returned by Invenio.

6.5.1 The Hybrid Solution Using the Present Search Engine of Invenio

Storing Semantic Annotation in Invenio

Section 6.3 described an extended data model allowing a more flexible description of documents. Part of the implemented extensions allows storing arbitrary non-MARC metadata describing different objects. In our model, figures are described in Invenio as documents. The general storage engine of the digital library system does not manage descriptions of entities described by parts of the documents like coordinate systems, data lines or legends.

Hierarchy-based Semantic Queries Using Invenio Indices

At the moment of writing, INSPIRE is not integrated with any search engine allowing the execution of semantic queries. As described earlier in this chapter, the internal search engine allows indexing the bibliographic records using forward and reversed keyword indices. The simple nature of those indices allows them to retrieve the search results efficiently. We believe that a wide range of figure-related applications can be implemented using this index. However, achieving the efficiency comparable to this offered by the execution of SPARQL queries would require additional data processing prior to the indexing and searching.

In this section we describe two techniques which can be used to implement a semantic search using the present Invenio search engine. One of them requires indexing a larger amount of metadata, but allows executing searches by using a single lookup in the index. In the second, a smaller amount of metadata needs to be indexed, but a larger number of index queries are necessary to retrieve results.

The techniques described in this section can be used to search for figures using a single annotation. They cannot be directly generalised to searches exploiting the graph structure of the semantic annotation.

Direct Indexing of Annotations

When indexing the semantic annotation using the internal Invenio techniques, it is better not to use the human readable textual labels. Chapter 5 showed that the interpretation of such labels as concepts from the ontology can usually be ambiguous. Instead, we should use some type of unique and persistent identifiers of the entities. The role of such identifiers can be played by the URIs used inside the ontology.

The most direct approach to the indexing of the semantic annotation could involve selecting the predicate which needs to be used for search. Then, creating a MARC metadata field describing the predicate and include it in the record describing the indexed figure. The content of this field should contain the URI of the concept with which a figure element is executed. Subsequently, the created MARC metadata field can be indexed using standard Invenio techniques.

When searching for a figure annotated with a given concept (in a particular context which is defined by the predicate), first we need to determine the URI of the searched context inside the ontology. Ontology files tend to be small compared to the side of the entire corpus of figure annotations inside INSPIRE and thus, can be processed in-memory using RDF files. Having the URI, we can execute a standard index-based lookup for a given URI. The results set will consist of numbers of records which are annotated with the given concept.

Expansion of More General and More Specific Terms

The described method allows discovering only the specific concepts from the ontology. Earlier we showed how to increase the accuracy of search results by using the knowledge of the taxonomy of concepts. We can increase the taxonomy-awareness of the Invenio searches by expanding the indexed term with all the related taxonomy terms. Instead of indexing only the URI of the concept used in the annotation, we can calculate the MARC field by concatenating its URI together with URIs of the related concepts. Similarly as in the case of demonstrated SPARQL queries, we should index (and thus allow them to be searched) only the concepts marked as standalone.

The corpus of figures indexed using this method can be searched directly, without any special code wrapping the executions of the search procedure. As such, this method is very easy to integrate with the existing Invenio interface. However, major changes in the structure of the taxonomy used for the annotation require reindexing of the existing terms as the concepts related with the given one can change. The method presented in the following section allows searching for concepts in a quickly changing ontology.

The method described in this section does not evaluate how exact are the detected matches. Intuitively, records directly annotated with the searched term should be considered as matching better than those found using a related term. The following method can also be easily extended to favour the exact matches over matches with related terms, while still taking them into account.

Execution of multiple searches

An alternative method involves executing multiple Invenio searches every time when user wants to retrieve a plot. Similarly to the naive method described before, we index only URIs of the exact concepts used in the annotations. However, we use a different technique to retrieve the data.

Typically, the ontologies used for annotation are small enough to be efficiently processed using RDF files and their in-memory representations provided by one of the standard RDF processing libraries. This is in particular true for all the ontologies which we use to annotate figures. Whenever there is a need of finding figures annotated with a particular concept, the system should execute a number of exact-match queries (as described before), searching for the exact term of interest and subsequently repeating for all the related terms. The related terms should be determined before the execution of the search.

In the described scenario the results of the exact search can be treated as better results than those which were discovered using similar concepts. It is also much easier to adapt this schema to different strategies of determining the related concepts. However, this method requires much more programming work to wrap the existing search engine calls in the logic that would access the ontology and combine the retrieved results. We believe that this approach is the best option for the application in INSPIRE.

6.5.2 Integration With an External Semantic Search Engine

This section covers an integration of the results from the internal search engine implemented in Invenio and used in INSPIRE with the results from an external semantics-aware search engine. The efficient execution of SPARQL queries is currently a subject of an extensive research [28, 1] and there are solutions for implementing such a search. However, the existing software solutions that allow searching inside large distributed data triple databases, tend to have difficulties when executing more complex queries [28]. Thus, they seem not to be ready to be used in a system processing large amounts of the semantic annotation. In this section we treat an external search engine as a complete separate system which interfaces with INSPIRE using SPARQL.

We assume the efficient execution of queries in this external system and concentrate on the architecture of interpreting the user-provided queries which require results from two search systems simultaneously and on integrating the results from both in a single INSPIRE answer. There exists a one-to-one mapping between figures stored in the knowledge base and the INSPIRE records, which facilitates the integration of the external results.

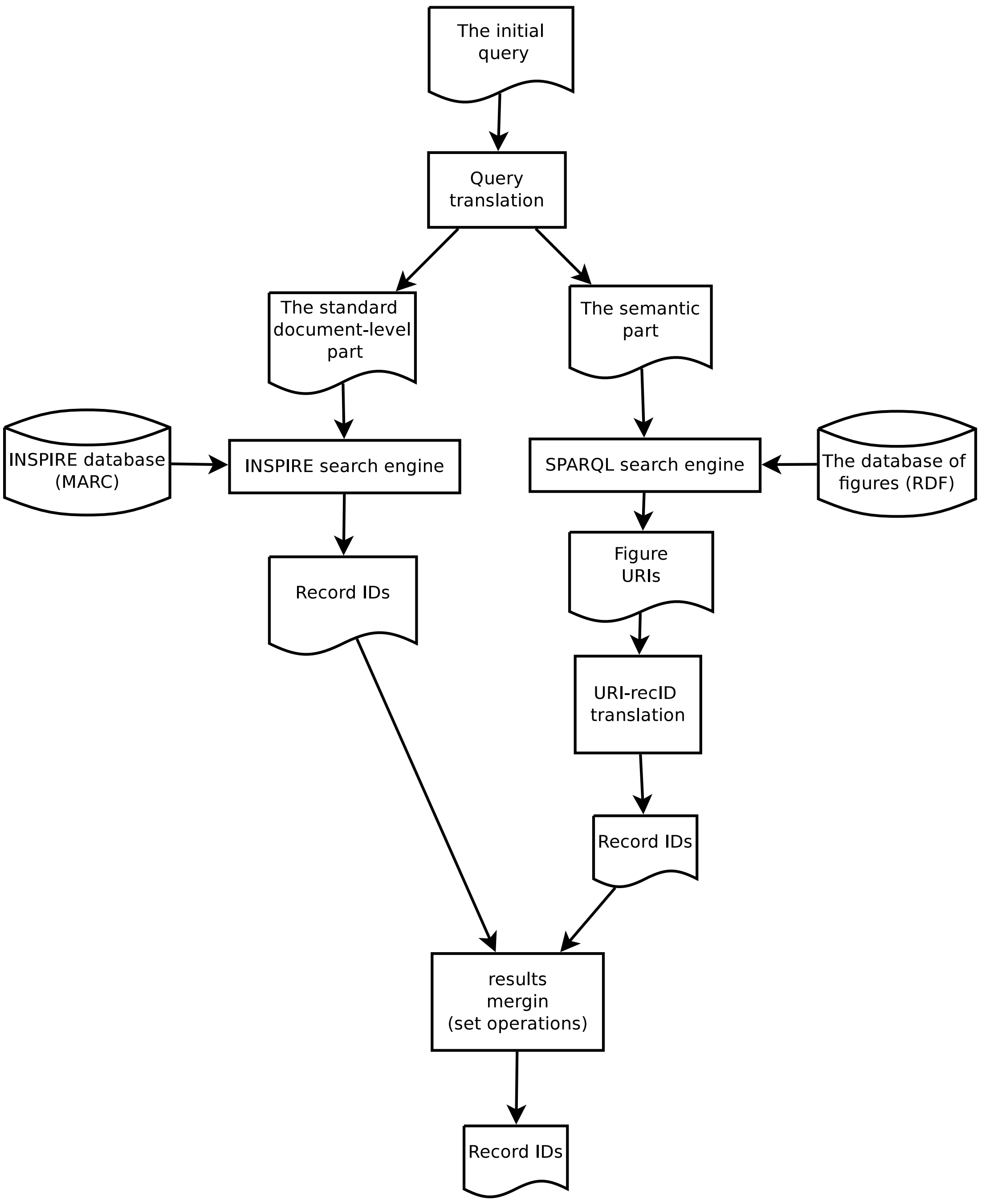


Figure 6.14: The process of merging results of a semantic SPARQL query with results coming from the INSPIRE search engine

Overview of the Integration

The most obvious use-case, when the semantic knowledge-base has to be queried is resolving a query which has been provided using a search box. Figure 6.14 depicts a possible integration of the existing search engine with a SPARQL endpoint. Every figure is described in both INSPIRE and in the RDF store, so both search engines are capable of providing results containing figures. With the introduction of a semantic search engine, every query will have to be processed by a module capable of splitting it into parts resolvable by the internal search engine and into a list of SPARQL queries which can be executed by the additional search engine. The internal search engine transforms every INSPIRE query into a set of matching record identifiers. Every semantic query would be resolved into a set of URIs corresponding to the figures. Subsequently, those URIs have to be translated into record identifiers of INSPIRE. All the record sets have to be merged into a single result set. This can be achieved by an application of set operations reflecting the structure of the original query.

The Translation to SPARQL

Translating queries between different query languages is a very broad topic. The present considerations are limited to a very minimal subset of cases which have the potential of being the most useful for the final users of our system. If the future needs show the insufficiency of the current approach, the method can be extended. The internal search engine of Invenio bases on the notion of metadata fields. The user can specify a meta-data field which should be searched for the desired keyword. Users are also given the possibility of intersecting and joining the result sets coming from different parts of a query. The metadata fields can be specified either by providing the MARC code or by an easier to remember textual name, which in turn gets translated to MARC. For example *title:* is translated to *245:*. In order to integrate the semantic figure search with the rest of the Invenio search engine, one can define a virtual metadata figure field:. When resolving the query, this field should not trigger a standard search inside a MARC metadata. Instead, a series of SPARQL queries should be executed.

In addition, INSPIRE can be extended with a list of predefined SPARQL queries searching for the most common figure types (including those for which the traditional storage has been extended). Each of these queries should be assigned with a keyword identifier. When resolving the content of the *figure:* field, each searched keyword would be resolved as the corresponding SPARQL query. In the case of encountering many keywords, they results should be merged in a similar way as it is done by Invenio for standard subqueries.

The Internal Usage of the Predefined Queries

The predefined queries known to INSPIRE can be used in a much wider context than during the translation from the INSPIRE query language. When displaying the results of some figures search (regardless if obtained using the internal search engine or from the semantic repository), the system can divide results into categories corresponding to predefined SPARQL queries. Such an approach, connected with the usage of the graphical user interface to allow the user to manipulate the displayed dataset is known as faceted search. Faceted search is widely used in both general-purpose search engines and is implemented in Invenio.

Applications of Comparing Figures

Chapter 5 described the construction of a similarity measure for comparing figures. Having a method of comparing figures based on their content can allow many interesting applications in INSPIRE.

Calculating the similarity between all pairs of figures stored in INSPIRE could allow discovering similarities between publications. Publications which include figures that illustrate similar processes or measurements are likely to be similar themselves. Such a system would discover similarities on a much deeper level than only using keywords.

User browsing a figure managed by INSPIRE could be also presented with a list of figures whose distance to the browsed one is small enough. Those figures could serve as recommendations of content which could be potentially interesting for the user.

Much additional work needs to be done to explore the methods of efficiently calculating distances inside the large corpus of INSPIRE and using the results as a part of the production system.

6.6 Integration of HepData

Thanks to the HepData project, physicists specialising in HEP have access to a database of datasets, which has been manually curated by a thorough review of scholarly publications over the years. HepData has been a standalone system, only being linked from external services like SPIRES. These links were being updated manually by the persons maintaining HepData, which was a time consuming process leading to a higher possibility of producing errors. The body of data stored in HepData consists of over 50.000 datasets coming from tables and figures and also includes additional datasets provided by the authors. INSPIRE aims at providing seamless access to the main scholarly artifacts in HEP. Thus, the content of HepData has been made accessible via the INSPIRE services as well.

Integrating the datasets in INSPIRE was also an opportunity to improve and redesign certain aspects of the user experience related to browsing the datasets. We implemented a new, more modern, method of rendering the metadata related to data tables. We also created an infrastructure for assigning persistent identifiers (Digital Object Identifier - DOI), which allow identifying the datasets outside the context of a publication and enable the possibility of data citations.

6.6.1 Establishing the Interoperability Between Systems

Architecture of the Integration

There existed several important requirements lying at the foundation of the interoperability of INSPIRE and HepData.

INSPIRE aims at being a complete information resource for HEP and wants to provide unique identifiers of data, allowing data to be cited. However, HepData provided only limited searching capabilities which could not be extended without having more extensive metadata about not only datasets but also publications. We wanted to allow HepData to use the annotations from INSPIRE to improve its searching capabilities.

We did not want to make one system completely replace the functionality of the other. Instead, we decided to establish a synchronisation process allowing both systems to use the same resources. At the same time we did not want to increase the amount of manual actions necessary on any of the sides and we wanted to keep strict separation of responsibilities in the same state as before the integration. This means that metadata describing datasets was to be curated by the HepData team and the metadata about publications by the INSPIRE team.

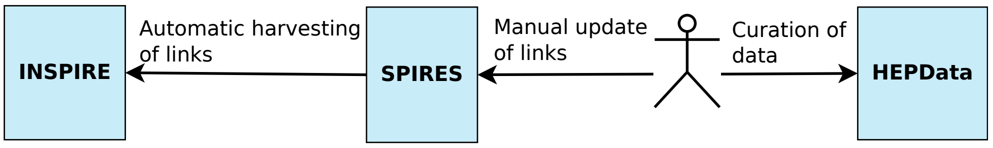


Figure 6.15: Interaction between HepData and SPIRES/INSPIRE prior to the integration.

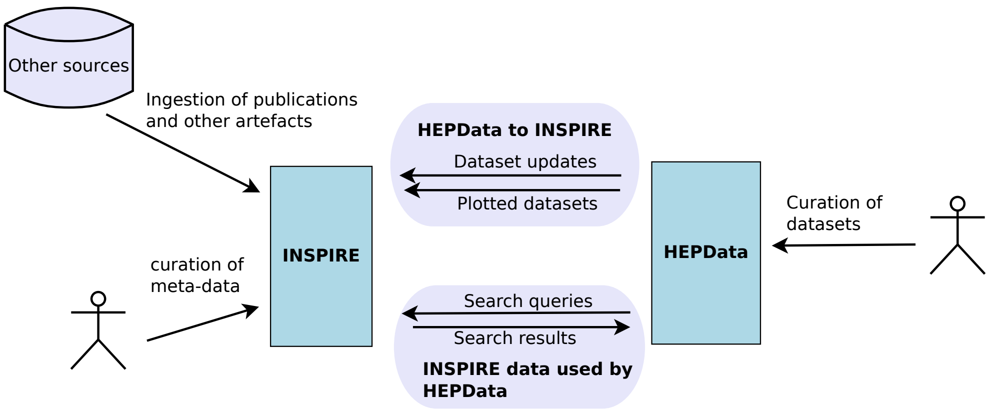


Figure 6.16: Relations between the two integrated systems.

This requirement forced us to address challenges related to maintaining both systems up to date after the data has changed in one of them. We also had to establish data exchange methods allowing the transfer of datasets between INSPIRE and HepData. Before the integration, there was no automatic communication between systems. Links to datasets were manually curated inside SPIRES (and subsequently, after replacing SPIRES, integrated in INSPIRE). The relation between both systems before the integration is depicted in Figure 6.15, while Figure 6.16 shows interactions between INSPIRE and HepData after being integrated.

The amount and the nature of information transferred in both directions are not symmetrical. We have decided to store a complete content of HepData on the side of INSPIRE, which allowed us to integrate metadata in the internal search engine and by this, extend the capabilities of searching for the datasets. Having a complete copy of the data allowed us also to assign data identifiers in a persistent way. Keeping the data and publications at a single location will in the future simplify the analysis of users behaviour, which in turn can lead to the improvement of the quality of service. In addition, INSPIRE is under an active development which makes integration of the data inside it easier. An alternative would be to extend HepData with APIs for manipulating and accessing data and making INSPIRE capable of using these. Such a process would be much more complicated and resource consuming.

The data from INSPIRE is not stored on the HepData side. Instead, we have decided to exploit the INSPIRE search engine directly every time HepData service wants to query the INSPIRE dataset. This has been dictated by the fact that INSPIRE implements a highly efficient search engine operating on separate fields of publications metadata and full text. The following subsections discuss the decisions related to the exchange of information between systems and the storage of figures inside INSPIRE.

The Ingestion of HepData into INSPIRE

At the moment of writing, after almost 40 years of existence, the number of publications having data attached to them is close to 7000 and the rate of increase is expected to be constant because all the datasets are processed manually before being made publicly available. Retrieving and processing such a number of records can be performed very quickly using modern computers, so scalability was not an issue when designing the integration of INSPIRE with HepData. We have decided to implement a simplistic solution in which we always harvest a complete set of HepData entries. In order to achieve this model of interoperability, we had to extend HepData with the capability of exporting the list of all present datasets, which was not possible prior to the integration. If the size or growth rate of HepData was much larger, the scalability could be achieved by designing more complicated mechanisms to retrieve only data changes applied after previous synchronisation.

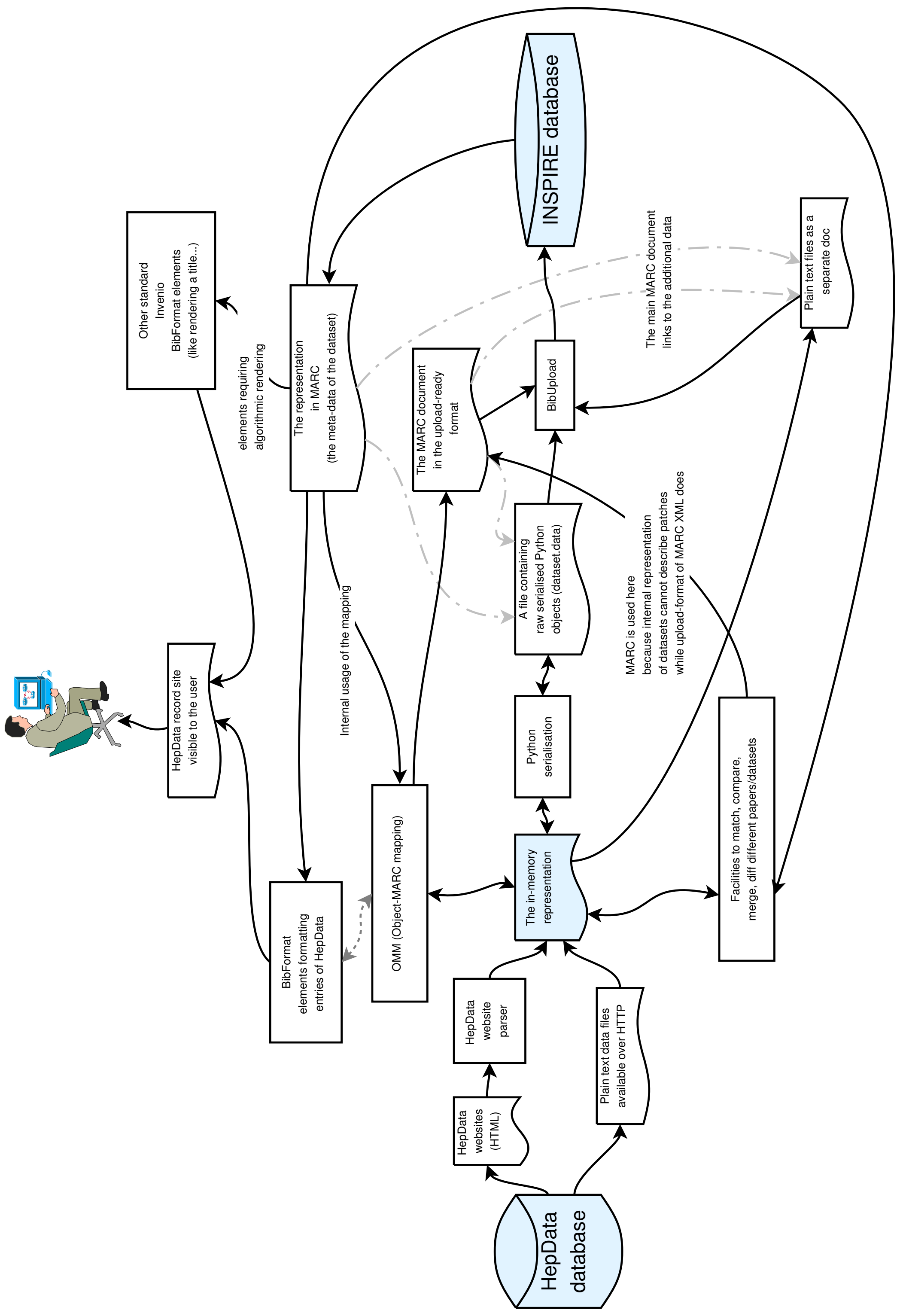

INSPIRE database
Other standard Invenio BibFormat elements (like rendering a title...)
elements requiring algorithmic rendering
The representation in MARC (the meta-data of the dataset)
The MARC document in the upload-ready format
BibUpload
The main MARC document links to the additional data
Plain text files as a separate doc
A file containing raw serialised Python objects (dataset.data)
MARC is used here because internal representation of datasets cannot describe patches while upload-format of MARC XML does
Internal usage of the mapping
HepData record site visible to the user
Python serialisation
Facilities to match, compare, merge, diff different papers/datasets
The in-memory representation
OMM (Object-MARC mapping)
BibFormat elements formatting entries of HepData
HepData website parser
Plain text data files available over HTTP
HepData websites (HTML)
HepData database

Figure 6.17: The complete flow of data between the HepData system and different parts of Invenio

During the development of the integration of both systems, we have extended HepData to store INSPIRE identifiers of the publications in connection with the data records. This extension has allowed HepData users to address its content using INSPIRE identifiers as parts of the URLs. When INSPIRE needs to retrieve HepData related to a particular publication record, it uses the INSPIRE record number to build the URL of the HepData system representing all the data attached to a single publication. The document stored under the URL is retrieved and its content parsed to extract the described data. Harvesting of the data could have been done using a dedicated protocol similar to OAI-PMH [17][52], which is fully implemented in Invenio. However, for the purpose of the prototype, we decided to use a much more simplistic approach by parsing the original HTML page and retrieving data from its content.

INSPIRE executes The HepData harvesting procedure periodically. Before every harvesting, INSPIRE already contains records describing datasets acquired during previous executions. Datasets described by these records are also retrieved during every subsequent execution of the harvesting procedure. This creates a need for determining which harvested datasets correspond to which already existing records. The matching (and merging) with existing datasets happens in the scope of every publication separately. It has to be done using the data of the datasets because HepData does not maintain references to the internal dataset identifiers in INSPIRE and the internal HepData identifiers (position within a record) are not persistent. We detect which datasets stored in INSPIRE have changed in HepData, which have remained the same, and which have been added or removed. These changes are used to generate a minimal patch that has to be applied to the existing records.

Figure 6.17 shows a complete workflow of data between Invenio components and the HepData system.

Storage of HepData in INSPIRE

Section 6.3 describes different types of objects which can be stores in an Invenio-based repository like INSPIRE. The metadata of the bibliographic records is described using the MARC[9] format and additional objects together with their accompanying metadata can be described using additional *BibDoc* structures.

In order to make HepData an integral part of the INSPIRE content, harvested datasets are transformed into records stored in a MARC format similar to the one used for regular publications, encoded with the format depicted in Figure 6.20. A separate MARC record is created for every data table extracted from a HepData system and is linked to the record of the main publication using the 786 MARC field. All records describing datasets are added to a new INSPIRE records collection called DATA. An example of a Full MARC record describing a HepData dataset can be seen in Figure 6.19. Figure 6.18 shows how the entries of HepData

[9]`https://twiki.cern.ch/twiki/bin/view/INSPIRE/DevelopmentRecordMarkup`

are seen by the Invenio code manipulating the data.

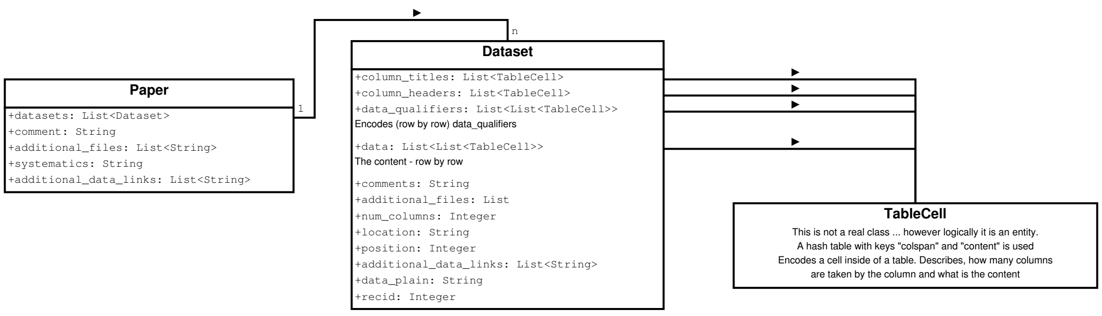

Paper
+datasets: List<Dataset>
+comment: String
+additional_files: List<String>
+systematics: String
+additional_data_links: List<String>
1
n
Dataset
+column_titles: List<TableCell>
+column_headers: List<TableCell>
+data_qualifiers: List<List<TableCell>>
Encodes (row by row) data_qualifiers
+data: List<List<TableCell>>
The content - row by row
+comments: String
+additional_files: List
+num_columns: Integer
+location: String
+position: Integer
+additional_data_links: List<String>
+data_plain: String
+recid: Integer
TableCell
This is not a real class ... however logically it is an entity.
A hash table with keys "colspan" and "content" is used
Encodes a cell inside of a table. Describes, how many columns
are taken by the column and what is the content

Figure 6.18: The simplified structure of entities used to encode HepData entries in the in-memory model in Invenio

MARC is a standard for describing metadata rather than the data itself. In the records corresponding to data tables we store only bits of information which can be considered metadata. This includes the global textual description of the dataset, headers and titles of data columns, descriptors of data and location of data inside the original publication.

The main corpus of data, with few exceptions, consists of a series of numbers that should be displayed inside a table. HepData facilitates the reuse of the data by allowing the download in several popular formats in addition to displaying them as web-page content. The harvesting script downloads only one format ("plain text") and allows its retrieval directly from INSPIRE. Rather than storing data inside a MARC record, we store the attached data using the BibDoc infrastructure.

```
001155413 001__ 1155413
001155413 245__ $$9HEPDATA$$aKEK. MEASUREMENT OF POLARIZATIONS FOR K+ N ELASTIC AND...
001155413 336__ $$tDATASET
001155413 520__ $$9HEPDATA
001155413 6531_ $$c1$$rK+ N --&gt; K+ N
001155413 6531_ $$c1$$kSQRT(S)$$v1.819 GeV
001155413 786__ $$hT 1.$$q1$$w179758
001155413 8564_ $$uhttp://inspirehep.net/record/1155413/files/Data.plain$$ydata of the table
001155413 910__ $$dCOS(THETA(RF=CM))$$n0$$t
001155413 910__ $$dPOL$$n1$$tPLAB : 1.06 GeV c^-1
001155413 980__ $$aDATA
```

Figure 6.19: A complete MARC record describing a data table

field	subfield	meaning
245__	a	The HepData-assigned description of the dataset
	9	Always set to HEPDATA. Designs provenance of this information
520__	9	Always set to HEPDATA. Means the provenance of the record
710__	g	The collaboration from which the dataset comes (Inherited from the main publication record)
786__		The link to the main publication record
	w	The identifier of the main publication record to which the current dataset is related
	r	The arXiv identifier (inherited from the 037__a field of the main record
	h	The location of a dataset inside the original publication
336__	a	Type of the dataset (FIGURE or TABLE)
980__	a	The collection to which the record belongs, always DATA
6531_		Describes data qualifiers
	r	If the qualifier described a reaction, this subfield encodes its description
	k	A type of qualifier (in the case of being different than reaction)
	v	A description of a qualifier (in the case of being different than reaction)
	c	repeatable subfield encoding which columns are described by a qualifier
	9	Always set to HEPDATA
910__		Description of subsequent data columns
	d	A column description (appears below data qualifiers and above the data)
	t	A more descriptive column title
	n	A number of the described column. Starting from 0 for the first x column
8564_		Links to additional data (not coming from the publication)
	u	The URL
	y	The text shortly describing linked dataset
	3	Always set to “ADDITIONAL HEPDATA”

Figure 6.20: The syntax of MARC format used to describe HepData entries

6.6.2 Possibilities Created by the Integration

Integrating HepData with INSPIRE paves the way to many new possibilities in the data management and the search. This section discusses added values of the integration, in particular for the research community. Additionally, while integrating the two systems, we have made a number of changes which considerably improved the user experience but are not directly related to having data and functionalities shared between two systems.

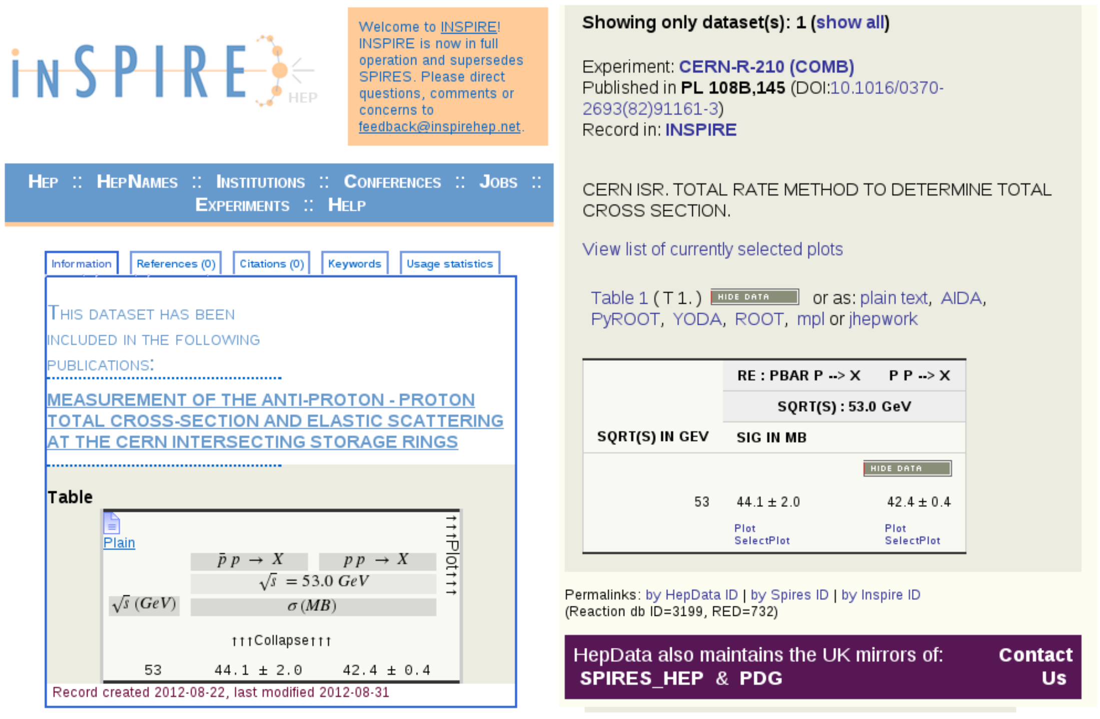
Easier Discovery of the Data

INSPIRE is becoming a single point of entry for accessing different types of scholarly content. The storage of HepData tables as separate MARC records allowed indexing them using standard Invenio techniques. As a consequence, formulating a single search phrase in INSPIRE can now lead to a discovery of not only articles, but also HepData records. This increases the visibility of HepData and, as datasets correspond to parts of articles, also the granularity of search in INSPIRE.

The visibility of HepData datasets has also increased because they are now an integral part of the display of a standard INSPIRE record. When displaying information about a publication, the user is presented with an additional tab showing related HepData. Also when a user browses figures, a link to the underlying HepData entries can be displayed and datasets can be accessed by making only a single mouse click.

Display of Plots and Data Qualifiers

The original HepData site provides powerful capabilities of plotting data included in tables. This functionality allows combining in the same plot data lines coming from different tables or even publications, which makes comparing results much easier.



Welcome to INSPIRE! INSPIRE is now in full operation and supersedes SPIRES. Please direct questions, comments or concerns to feedback@inspirehep.net.

HEP :: HEPNAMES :: INSTITUTIONS :: CONFERENCES :: JOBS :: EXPERIMENTS :: HELP

Information | References (0) | Citations (0) | Keywords | Usage statistics

THIS DATASET HAS BEEN INCLUDED IN THE FOLLOWING PUBLICATIONS:

MEASUREMENT OF THE ANTI-PROTON - PROTON TOTAL CROSS-SECTION AND ELASTIC SCATTERING AT THE CERN INTERSECTING STORAGE RINGS

Table

Plain

	$\bar{p}\,p \to X$	$p\,p \to X$
	$\sqrt{s}\ = 53.0\ GeV$	
$\sqrt{s}\ (GeV)$	$\sigma\ (MB)$	
53	44.1 ± 2.0	42.4 ± 0.4

↑↑↑Collapse↑↑↑

↓↓↓Plot↓↓↓

Record created 2012-08-22, last modified 2012-08-31

Showing only dataset(s): 1 (show all)

Experiment: CERN-R-210 (COMB)
Published in PL 108B,145 (DOI:10.1016/0370-2693(82)91161-3)
Record in: INSPIRE

CERN ISR. TOTAL RATE METHOD TO DETERMINE TOTAL CROSS SECTION.

View list of currently selected plots

Table 1 (T 1.) HIDE DATA or as: plain text, AIDA, PyROOT, YODA, ROOT, mpl or jhepwork

	RE : PBAR P --> X	P P --> X
	SQRT(S) : 53.0 GeV	
SQRT(S) IN GEV	SIG IN MB	
		HIDE DATA
53	44.1 ± 2.0	42.4 ± 0.4
	Plot SelectPlot	Plot SelectPlot

Permalinks: by HepData ID | by Spires ID | by Inspire ID
(Reaction db ID=3199, RED=732)

HepData also maintains the UK mirrors of: SPIRES_HEP & PDG   Contact Us

Figure 6.21: Comparison of how the data is displayed in INSPIRE and in HepData

These capabilities have been integrated in the INSPIRE view of data. As can be seen in Figure 6.21, when displaying a data table, the user first sees only the most general descriptions of data. Two expanding options are available and they can be used to get more insight into data. The user can expand the display in the graphical direction (to the right) and to the direction of details (downwards). Being in the most general descriptions mode and expanding the graphical aspect leads to displaying a general plot containing all data lines present in the table. Expanding down allows seeing the numbers and, if already being in the plotting mode, separate plots for every data line. Hovering over a plot highlights the data column used to generate the display. Clicking on a plot redirects to a HepData page which allows modifying the display parameters (like types of scales, ranges etc...) or add/remove data lines. When the user displays the INSPIRE page showing HepData, the plots are generated on the fly by a Java application running on the HepData server. Thus, this is another example of reusing the services from one platform (HepData) by the other platform (INSPIRE).

A very important part of the description of data stored in HepData are the headers indicating what type of physical process is being described by the data. Figure 6.21 depicts an example of these descriptors presented in the old HepData system. The description includes the type of a process, but also the numerical constraints specifying the conditions under which an experiment has been carried. The data headers have a form of mathematical fomulæ. As can be seen in Figure 6.21, the mathematical language used a special text-based notation. Nowadays, as there are various systems allowing the display, storage and processing of mathematical formulæ, the text-based language becomes outdated.

As a part of the HepData–INSPIRE integration, we developed a system for translating formulas described in the old format to $\LaTeX$. The translated version can be displayed in the browser using the MathJax javascript plugin.

Making Data Citable

There is an emerging need within the HEP community to track potential reuse of shared research data. Similar tracking is achieved for scholarly papers by registering the network of citations between the publications. INSPIRE along with other document repositories, extract references present in the publications and uses the graph to calculate statistics of popularity of different publications and authors. This is possible because there exists a semi-formal code defining, how one publication should be referenced from within of another. Publications can be identified by the place (journal, conference etc . . .), where it has been published. These places typically require that the proposed paper has not yet been published (or is not considered for a publication) in a different place. However, such formal rules for datasets do not exist. There exists some initiatives gathering scientific datasets, but they are not uniform and so, there is no uniform manner of addressing the resources gathered within them.

On the other hand, Digital Object Identifiers (DOI)[10] provide a system that allows identifying any type of digital objects. We have decided to use this system in order to provide identifiers for the datasets managed by INSPIRE. DOIs can be cited in the publications (and other types of resources) which use datasets. They are persistent and supposed to be unique (one identifier per object). These properties make them a good corner stone of a system that allows tracking the usages of scientific data. DOIs assigned by INSPIRE will be received from the DataCite consortium[11].

The corpus of HepData comes from data associated with scholarly publications: part of the data appears inside the original papers explicitly; and another part is supplementary and was used to prepare plots or to derive results. Only datasets that are not explicitly present in the publications should have DOIs assigned; the others can be uniquely identified using the parent publication identifier. HepData stores the exact provenance of the data, which allows us to

[10] `http://www.doi.org/`

[11] http://datacite.org/

automatically identify the additional data.

As mentioned earlier, the harvesting procedure considers situations when datasets have to be updated. This might seem to be incompatible with permanent identification. However, the nature of scientific publications simplifies the situation. After an article or a book is published, it is considered to be in a final version which cannot be modified. The same property is inherited by associated datasets. This implies that the only possible updates are corrections of mistakes related to the extraction or description of the data. As such, updated datasets describe the same data and the updating does not interfere with the DOI assignment procedure.

6.7 Summary

This chapter described the practical aspects of the figures processing in the context of a large-scale digital library (INSPIRE). We provided an introduction to the Invenio system and to the INSPIRE digital library, which is based on this software platform.

In Section 6.4 we presented the figure ingestion workflows of INSPIRE, taking into account different possible sources of figures and their annotations. The previously existent extractor allowing the extraction of figures from LATEXsources of the publications has been integrated with the PDF extractor in the process of matching and merging the descriptions. The automatically extracted (meta)data can be manually corrected using a manual tool, whose prototype we have developed.

As a part of the integration, we have extended the data model allowing the description of objects in INSPIRE. We described the extensions which we made to the internal data model allowing the storage of non-record objects in INSPIRE. The new data model extends the subsystem of storing documents by adding the capabilities of describing connections between documents and by making the abstraction describing figures more modular. The object of the previously and the newly added types can now be described with custom non-MARC metadata. All the created extensions are fully managed by Invenio and updates to the non-MARC metadata (the *MoreInfo* structures) are done by the *BibUpload* module. This assures both the data consistency in the presence of multiple concurrent updates and also a higher integrity of the entire Invenio system from the architectural point of view. The chapter continued with the description of how figures can be described and stored in an Invenio-based repository by using the new, extended data model.

Chapter 5 described how we can search for figures using their semantic annotation and the RDF database supporting SPARQL queries. This chapter elaborated on the methods of indexing figures in Invenio, which does not have an integrated semantic repository. We presented two approaches. The first, more pragmatic yet less powerful one, allows indexing certain views of the semantic annotations using the present search engine. The second, allowing the integration of an external search engine and merging its results with those retrieved from Invenio. We concentrated on the process of managing SPARQL queries as a similar integration has already been done with Solr search engine and the technical aspects of integrating a SPARQL search library would remain the same.

In this chapter we also described our experiments with integrating different types of publication artefacts inside the INSPIRE repository. The subject of the work consisted of the datasets managed in the HepData project. At present, those datasets are available as full records in INSPIRE.

The present work[12] has improved the possibilities of INSPIRE and the Invenio platform in general. However, it is only the first step towards building a system capable of fully using the potential of semantically-annotated figures. Much work needs to be done to finish all the aspects of the integration. There are also many new directions of the development, which can be explored in the course of development.

The present method of the ingestion of figures takes into account two main sources of publications. LaTeXfiles coming from arXiv.org and PDF documents. The process of matching figures from different sources is based on the extracted text of caption. A potentially better method could first try to match using the location of figures and then, if location from the LaTeXfile cannot be determined, should fallback to the present method of matching using the captions. Determining locations based on the figure and the PDF document could be implemented using various techniques of searching one image inside another.

The prototype of the manual figures extractor tool described in Section 6.4.3 demonstrates the possibility of creating intuitive tools to select figures inside the text of publications. However, the present amount of metadata which can be selected using this tool is limited. The future work within the scope of the INSPIRE project, could enable the manual extraction tool not only to serve as the method of the extraction of figures. The tool could be extended with the interface allowing the semantic annotation. Users of the manual extractor could be allowed specifying concepts from the HEP ontology, which describe the figure They could also be allowed manually establishing links between different figures already managed in INSPIRE and between figures and other artefacts.

The manual description of the internal structure of figures could be also done using the manual annotation interface. As described in Chapter 2, describing the semantic of a figure is a long process requiring many similar annotation elements. Using a graphical interface to select every single element of this annotation separately, would be a laborious and rather boring task. Chapter 4 showed the process of automatic meaning extraction, where simpler properties were used to detect those more complicated. Typically, the failure of extracting content from a particular figure is caused by failure in one or more of these independent steps. The manual annotation interface could structure the user input in a similar manner as the steps of the automatic algorithm.

[12]`https://github.com/ppiotr/Invenio/tree/current`

Initially, the interface could display all the annotation automatically generated by the automatic algorithm. Users could be given the possibility of modifying the annotation elements. Whenever the data changes as a result of a manual modification, the automatic semantics extraction algorithm could be reexecuted. In the course of the execution, parts of the annotation, which have been modified by the user, could substitute those computed automatically. This would in turn modify the way in which the derivative properties are calculated. As a result, corrections in the erroneous annotation of low-level elements could cause automatic correction of the more high-level annotation.

An example scenario of such corrections could look as follows: The description of the data generated by the automatic extractor is incorrect because a wrong pair of lines has been selected as axes of a coordinates system. The user is presented with an interface showing the figure with detected axes. Users see the error and select the correct lines of the axes. The extraction algorithm is executed with and additional input of the metadata provided by the user (this consists of two bits of information: that the previously detected pair of lines is not a coordinates system and that another pair of lines is the real coordinates system). After the step of detecting the coordinate systems, the intermediate results are revised using the correct user-provided metadata. The subsequent steps of the algorithm have access to correct input and thus, are able to correctly detect ticks on the correct axes. After the automatic extractor runs again in the background, the annotation displayed in the interface is updated and user if given a chance of additional editing, which in turn can cause more recalculations. We believe that such a method can converge to the correct metadata in a small number of steps, leading to a much smoother user experience and the need of a much smaller manual input.

The feedback provided by users using the annotation interface, could also be used to improve the training of the detection algorithm. Parts of the algorithm that uses machine learning techniques (like SVM) could use the provided correct data samples (like the manually annotated coordinate systems from the previous example), to improve the results of future extractions.

A much smaller work has to be done inside the manual figures extractor to fully integrate it with the authentication. At the moment, users of INSPIRE can claim that a publication has been written by them and to create their user profiles within INSPIRE. At the moment of writing, the authentication is achieved using the APIs of arXiv.org. Similar integration should be extended to the case of manual extraction of figures. Also, user accounts which can appear in INSPIRE in the future, should be used to authenticate users and to assign them to different roles in the process of figures selection.

The integration with HepData has already been deployed within INSPIRE and can be used by the scientists. However, there remain certain aspects which are subject of an ongoing work and which require further refinement.

The harvesting schema using a full dump of the HepData page has proven to be less effective than expected. First of all, harvesting large amounts of data from the HepData servers disturbs the normal functioning of the service. The harvesting script implemented within INSPIRE uses the technique of waiting for a predefined period of time after retrieving a dataset. This reduces the load generated on the HepData server, but also slows down the harvesting process. The slow speed of harvesting is also related to the fact that retrieving every dataset separately generates a high volume of small HyperText Transfer Protocol (HTTP) requests.

The present harvesting method uses human-readable HTML pages as the source of the data. Humans are very good at recognising the content regardless of the structure of the web page. However, computer scripts designed to parse the content are very vulnerable to certain types of changes made to the template of a web page. Modifications of HepData have already caused problems with erroneous harvesting on the INSPIRE side. A temporary measure consisted of freezing the view of data for the harvesting purposes (By making a copy of all the websites presenting data and leaving it without modifications). However, practice has shown that this was not a perfect solution (These pages have been changed anyway).

The present design has been a result of the requirement that we want to minimise the number of changes made to HepData. Over time, this requirement stopped playing an important role, which opened a way of improvements. At present, part of the INSPIRE and HepData team are working on establishing a protocol with two main purposes: retrieving only the modifications of the HepData corpus; and retrieving (meta)data of many datasets using a single request. Implementation of such a protocol will require extending HepData with a more precise tracing of what has changed. Also, an additional work will have to be done in order to implement the protocol itself on both INSPIRE and HepData sides. All the previously mentioned problems of retrieving datasets are expected to be solved by the introduction of this protocol.

Besides standard datasets, HepData maintains a large corpus of additional datasets uploaded by authors and not appearing directly inside the publications. These datasets are ignored by the present integration because all of them are annotated manually and do not have a common structure which could be used when encoding them in INSPIRE. INSPIRE as a preservation platform of HEP, would be interested in ingesting and managing also these datasets. However, much more work needs to be done because of the less regular structure of the data.

This thesis describes how figures can be annotated in a semantic manner. Similar annotation can be in the future extended to datasets. Such an extension could considerably increase the semantic interoperability between different systems and would allow compareing datasets with other artefacts (like figures and tables) using their internal data.

Chapter 7

Conclusions and Future Work

7.1 Research Contributions

This thesis has proposed a set of methods to facilitate the management of figures in scientific digital libraries. This management includes acquisition of figures, their annotation and searching facilities based on figures.

In Chapter 2 we described a HEP Figures Ontology (HFO) that defines the taxonomy of different types of figures as well as other elements participating in the scholarly publication process. The figures taxonomy has been created by analysing the types of figures which appear most frequently in HEP publications. The provided vocabulary allows describing all types of figures at a general level: the annotation with concepts of HEP; and the annotation of their relation with the publication in which they originally appear (e.g. caption, name or text references). The general annotation uses the Dublin Core vocabulary and extends it in the aspects which are specific for figures. The domain-specific concepts used in the annotation come from established physics-related vocabularies (HEP Taxonomy, Measurable Units Ontology). HFO ontology also contains the vocabulary which allows the description of plots (the most common type of HEP figures) with most of the detail of their internal semantics: the internal structure of plots and the annotation of its different elements. The meaning of these elements is dependent on the meaning of other elements. During the design of the ontology, we tried to minimise the impact of changing some elements of the annotation on the meaning of the others. The core description focuses on the aspects of a plot, which affect the understanding of the content, ignoring those being only elements of a particular visual design decision. According to our knowledge, prior to our work, there was not a formal vocabulary allowing the description of figures in a comprehensive manner. Our work on HFO ontology enabled us to describe figures in INSPIRE and to construct the searching facilities. Other disciplines containing figures can also take advantage of HFO to describe their figures or extend HFO minimally to define new types of figures. The main components of this ontology were presented in [42].

In Chapter 3 we described a novel approach of extracting graphs from PDF documents (where parts of the content are not annotated with the corresponding logical entities like figures or paragraphs), and to describe their publication-level metadata using the previously described vocabulary. All the existing tools enabling the extraction of graphics from the documents yielded very low recall rates for the scholarly documents. Our method of detecting fragments of PDF documents that correspond to figures is based on a series of observations on the character of publications and uses the techniques of computational geometry and the artificial intelligence: geometrical clustering; page segmentation and machine learning. During the extraction procedure, every page of the publication is treated as a set of graphical or textual primitives. The presented algorithm clusters those primitives and identifies sets corresponding to figures, tables and their captions. The tests have proved our method to be efficient during the tests on the corpus of HEP papers from INSPIRE. The plot extraction method has been published in [43].

Chapter 4 described a module for extracting semantics from vector-graphics representations of scientific graphs. We presented how the semantics extraction problem can be divided into separate modules. Additionally, we have demonstrated the feasibility of developing such modules by describing an extraction module capable of detecting coordinate systems inside figures. Similarly to the figure extraction, the semantics extraction first preprocesses the document by splitting all its elements into the smallest possible elements: 2-dimensional line segments and single-word portions of text. Other types of content (e.g. segments of curves and included raster images) are ignored. Subsequently, those primitives are used to detect the coordinate-system candidates. Those candidates can be either coordinate systems of a plot or false candidates. We have proposed the usage of Support Vector Machines (SVM) to distinguish between the true and false candidates. The detected coordinate systems consist of axes together with related elements like axis ticks, labels or labels of the axis ticks. The created annotation carries sufficient information to interpret the remaining part of the plot. At the sample place of the thesis we have also demonstrated, how the metadata produced using the extraction module could be encoded in RDF using the HFO ontology.

Chapter 5 describes the usage of the previously created annotation in order to search for figures in a semantic manner using the SPARQL query language and other semantic technologies. We presented a prototype allowing the retrieval of figures based on the physics concepts with which they have been annotated. The tests on a sample corpus of annotated data have shown an increase of recall with respect to traditional text retrieval systems thanks to the use of the hierarchical relations in the proposed ontology. In the same chapter we also presented more complex queries exploiting the annotated structure of figures. The prototype of Chapter 4 does not provide a complete annotation to create a large dataset which could be used to test the presented queries. However, the usage of a smaller testing set has shown that the proposed annotation allows retrieving the relevant results.

Also related to the searching functionalities, at the end of Chapter 5 we presented a method of constructing a semantic similarity measure which can be used to determine which figures are similar. Unlike in the case of typical images similarity, where the similarity is constructed based on the basic image features like the pixel colours, the proposed similarity measure uses the semantic annotation to compare figures. Entities are compared using their internal structure and the direct annotation with concepts or literal values. When calculating the similarity between two entities consisting of smaller elements, the similarity is recursively computed between the constituent elements. The calculated similarities contribute to the similarity between higher-level objects according to a weight coefficient proportional to the importance of the type of the compared sub-elements. The similarity measure also takes into account the possibility that the annotation of one or more objects is not complete. Using a semantics-oriented measure allows detecting the similarity between figures which are visually different. The similarity measure can be useful to allow searching using examples of figures.

All the previously described components of a figure-processing infrastructure can be applied in the context of a particular information system. We partially integrated the prototype implementations with the INSPIRE digital library system. This work is described in Chapter 6 of the thesis:

- We have presented the ingestion workflow that allows including figures in INSPIRE. This workflow involves the usage of different extractors (at present allowing the acquisition of figures from LaTeXand PDF files), matching the obtained results and merging them in a way that maximises the advantages of both approaches. This ingestion workflow has been completely implemented in the course of the doctoral project.

- The new use case of persistently storing figures in INSPIRE required us to extend the document storage model of Invenio. The new data model adds capabilities of describing connections between stored documents and makes their storage more independent from bibliographical records describing publications. It also introduces a new method of storing more flexible metadata related to documents and to relations between them. A similar data model could be adopted by other systems using MARC as input/output format. We have also designed a system for coordinating the concurrent update of metadata in Invenio.

  Additionally, the chapter has described how to use the new data model to serialise the HFO annotation of figures. Relations between documents, which are the new type of object, allow for example relating the original document of a publication with the documents which are results of its processing and describing the publication context of a figure. The upgrade of the INSPIRE data model was published in [45].

- With respect to searching, chapter 5 described how we can search for figures using their semantic annotation and the RDF database supporting SPARQL queries. However, Chapter 6 elaborated on the methods of indexing figures in Invenio, which does not have an integrated semantic repository. We presented two approaches to solve this problem. The first, more pragmatic yet less powerful one, allows indexing certain views of the semantic annotations using the present search engine. In this approach, the predefined SPARQL queries could be translated into the constructs of the internal Invenio search engine. The second approach allows integrating an external search engine and to merging its results with those retrieved from Invenio. A similar integration has already been done with Solr search engine and many aspects of integrating a SPARQL search library could be solved in a similar manner.

- In Chapter 6 we have also described our experiments with integrating different types of publication artefacts inside the INSPIRE repository. The subject of the work consisted of the datasets managed in the HepData project. HepData is a repository of datasets which appear in the scholarly publications of HEP. Those datasets appear in the publication as figures, tables and sometimes are described inside the text. At present, those datasets are available as full records in INSPIRE. This integration was published in [44].

The software of the described prototypes, together with the software of the integration with Invenio, is available in different source code repositories hosted on GitHub[1] platform.

HEP figures are the main application domain of the work presented in this thesis. However, the majority of the results can be directly applied to the general case of figures. Small adjustments of the presented methods (e.g. replacing HEP taxonomy with different domain-specific vocabularies) can make the results applicable in different areas of human knowledge like biology or chemistry.

7.2 Future Work

This work contributes towards bridging the semantic gap stretched between the content available to be processed by the machines and this which is encoded in scientific graphs. This thesis has proposed the core design of a semantic-based figure management system, but it also opens the way for a vast variety of improvements, extensions and completely new applications.

First of all, figures are only one type of the objects that could be stored in a HEP digital library. Chapter 6 has described the integration of datasets coming from the HepData service. In the future, HFO could be extended to allow the annotation of those other types of objects with a more exhaustive level of detail. Such an extension could considerably increase the semantic interoperability between different systems and would allow comparing datasets with other artefacts (like figures and tables) using their internal data.

[1]`https://github.com/ppiotr`

With respect to the extraction of figures from the PDF documents, additional work is needed on heuristics allowing the correct detection of figures. Besides, additional refinements of the structure of the algorithm could improve the quality of the obtained results. The heuristics used by the algorithm are based on a number of numeric parameters which we have tried to optimise using automatic techniques. The results of the presented tests show a high recall and precision of the method, but additional extensions of the method can improve them even further. The method could also be reevaluated using a completely different set of publications, preferably coming from a different area of science. In addition to the evaluation of the efficiency of the method, it would be interesting to conduct a study of differences between the optimal algorithm parameters which should be used with a different corpus. Such a study could reveal which are the most important heuristics for different types of publications.

With respect to the architecture of the prototype we have developed for the extraction of figures, it is worth noting that it has a modular design, which allows its extension for different tasks. The internal metadata used at different stages at the extraction can be useful to extract different properties of the documents. As such, the prototype could be a basis for a more general PDF-data extraction framework.

Concerning the extraction of semantics from figures, it is necessary to extend the proposed methods. This thesis has described the prototype of a module extracting a single type of entity: coordinate systems. However, there are also other important parts of plots which must be extracted in order to use all the description capabilities of HFO and to enable a complete semantic figures search in INSPIRE. In a more distant future, the extraction method should be extended to allow also the extraction of the semantics of other types of figures like various types of diagrams.

When describing the semantic searching, we described a prototype system for concept-based queries and filtering figures according to their internal properties. However, additional work is necessary to identify other user search needs and to further evaluate (and improve) the searching process for HEP figures. The described similarity measure opens a wide range of applications. For example, this similarity measure could facilitate the clustering of a collection of publications according to the figure similarity. Additional effort is needed to implement this measure and to evaluate, how well it describes the logical similarities between figures.

Finally, further work is also necessary to advance in the integration of figures in the INSPIRE digital library:

- First, the work aiming at fully deploying the new data model and the extraction procedures inside the production system must be finished. Subsequently, all publications stored in INSPIRE should be processed and all the contained figures should be extracted and included in the INSPIRE collection. This will pave the road for the creation of the necessary indices necessary to search for figures. The implemented and described methods could be further improved. The method of matching and merging the annotations generated by different extractions can be improved by using also different criteria for comparing figures. The manual extraction tool can be extended to not only allow selecting figures within publications, but also describing their complete semantics. A more interactive scenario of the manual extraction could comprise execution of the automatic semantics extraction tool using the partial data coming from the user feedback. Also the statistics of usage and logs of the present annotation tool could prove to be useful when reevaluating the efficiency of the automatic extraction procedure. There was much work which lies outside the scope of the prototype, but is necessary for the successful operation in the production system. One of such details is the development of a complete user-authentication mechanism to assign user roles in the manual extraction tool. At present, every user is permitted to perform every possible action.

- Having the implementation of the similarity measure comparing plots will open a plethora of new possibilities for INSPIRE. Future work in this area could include the implementation of a content-recommendation system. INSPIRE should be able to suggest similar figures and publications to users browsing figures.

- The integration with HepData has already been deployed within INSPIRE and can be used by the scientists. However, certain aspects remain to be subject of an ongoing work and they will require a further refinement. The initial method of harvesting HepData datasets had been designed with the requirement of not making deep changes to the HepData system. After the initial prototype has been implemented, the method has proven not to be very efficient and the initial requirement decreased in its significance. At the moment, members of the INSPIRE and HepData teams work on establishing a new datasets harvesting method based on a dedicated protocol. Besides standard datasets, HepData maintains a large corpus of additional datasets uploaded by authors and not appearing directly inside the publications. These datasets are ignored by the present integration because all of them are annotated manually and do not have a common structure which could be used when encoding them in INSPIRE. INSPIRE as a preservation platform of HEP would be interested in ingesting and managing also these datasets. However, much more work needs to be done because of the less regular structure of the data.

Appendix A

The Most Common Types of Figures

This appendix presents descriptions and examples of the most popular figure types in HEP. It is intended as a catalogue of the figure types supplementary to the taxonomy description included in Chapter 2. The list is not complete and include only the types of figures that appear the most frequently in the publications stored in INSPIRE. In many cases the classification is ambiguous as figures might contain features of different types.

A.1 Plots

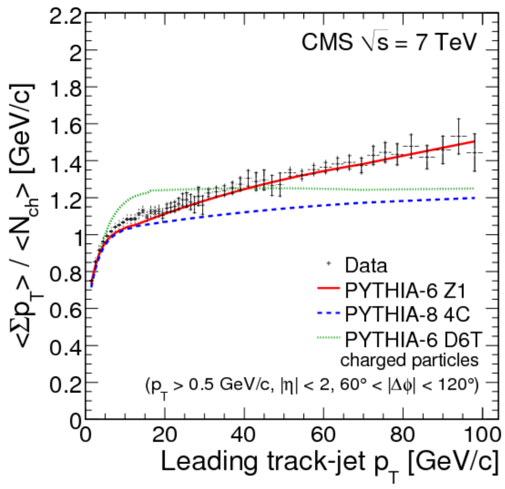
Plots can be characterised a graphics where a subarea of the canvas can be interpreted as a subset of some abstract space (the representation space). In the case of the majority of plots, the space into which the canvas points are mapped is $\mathbb{R}^2$; however different possibilities, like $\mathbb{C}$ or surfaces of manifolds are not uncommon.

Different types of plots are characterised by different usages of the mapped space or simply by a different interpretation of the encoded data. In some cases for example, the difference between generic measurement plots and histograms is very small and impossible to be decided using only the visual representation. The mapping between the space of the canvas and the abstract represented space is typically represented in a graphical form using the coordinate systems.

A.1.1 Generic Function or Measurement Plot

This is the most general type of a plot depicting the dependence of a function or measurement on a certain parameter. In the case of 2-dimensional function plots (plots of functions mapping subsets of 1-dimensional space into subsets of another 1-dimensional space), one of the dimensions typically depicts the value of the parameter and the other depicts the evaluated value of a function or the number obtained from a measurement. Most of other types of plots are a subclass of this one. If the value depends on N parameters, plot of this type can be drawn in N+1 dimensions.

Sometimes, more than one function/measurement is presented using a single figure. Usually in such a case, different colours are used to differentiate between separate data lines and the representation space is shared between them.



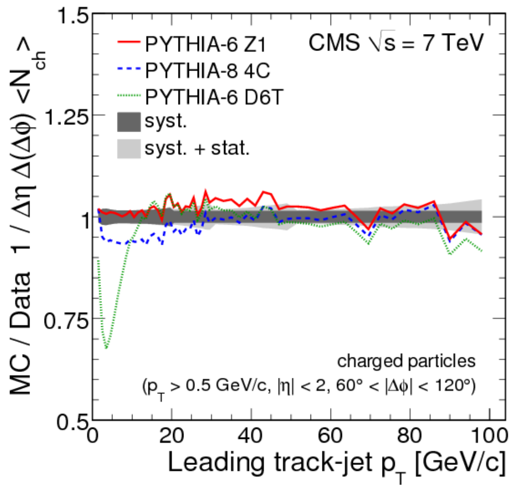


Figure A.1: Examples of function and measurement plots coming from publications stored at arXiv.org

A.1.2 Area Plot

The role of area plots is to indicate subsets of the representation space. Different subsets are typically represented by subareas of different colour.

Area plots can be for example used to represent regions in the phase space, where certain phenomenon appears or which preserves certain type of property. Examples of area plots can be seen in Figures A.2,A.3 and A.4. Exclusion plots are a particular type of area plots.

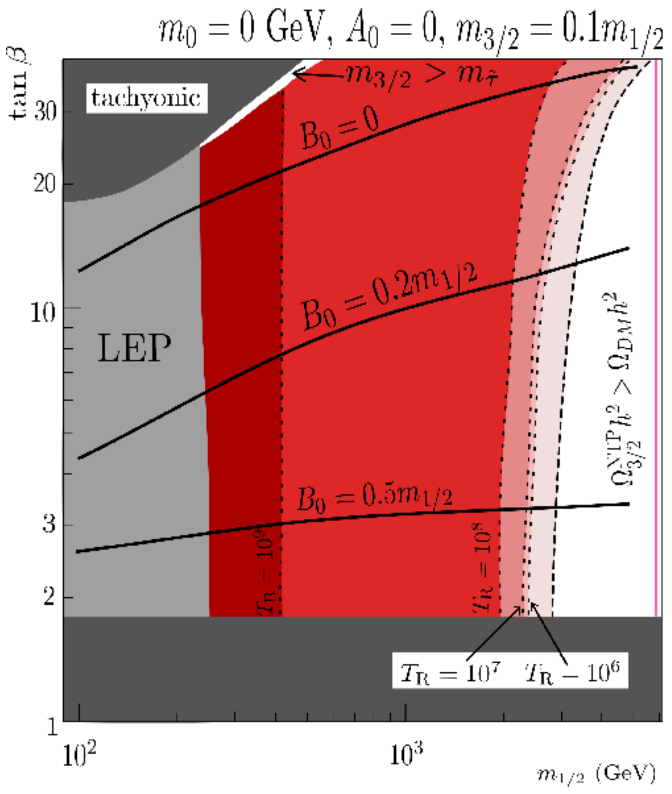


Figure A.2: Example of an area plot coming from publications stored at arXiv.org

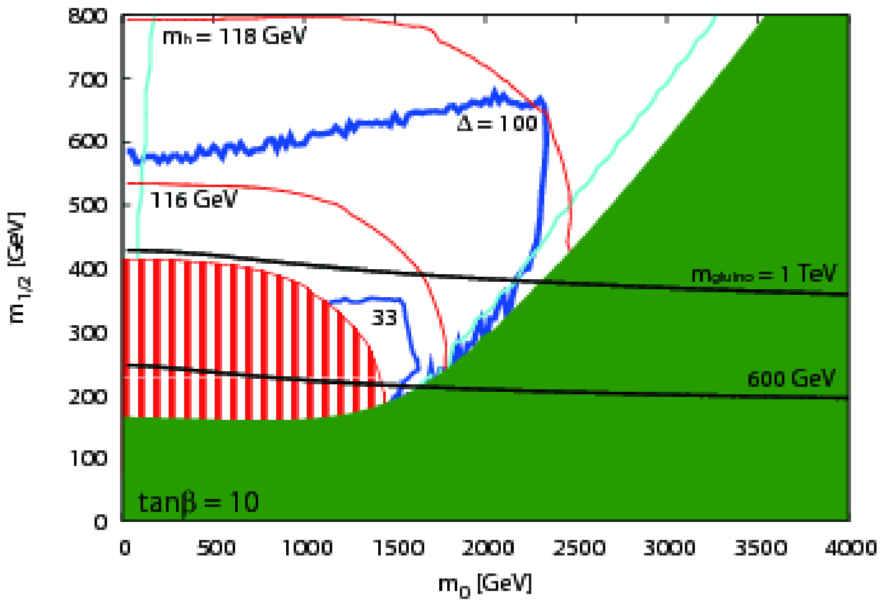


Figure A.3: Example of an area plot coming from publications stored at arXiv.org

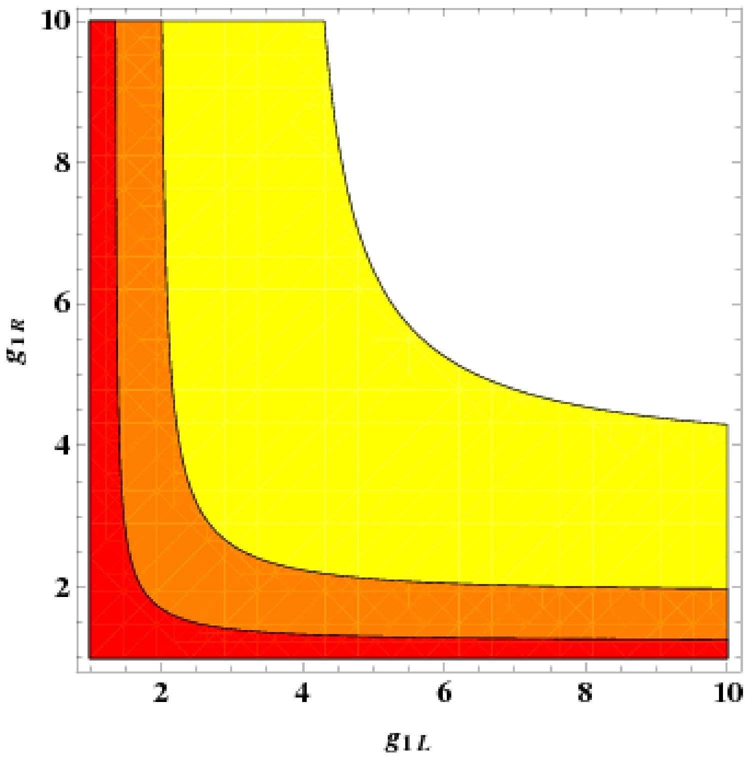


Figure A.4: Example of an area plot coming from publications stored at arXiv.org

A.1.3 Histogram

Histograms represent the aggregated numbers of occurrences of certain objects. Typically, histograms are 2-dimensional plots where one dimension in the representation space can be interpreted as the type of an object and the other as the number of appearances. In many applications, the objects which are counted are numbers or subsets of real numbers (e.g. intervals with a centre in a certain value).

Histograms can be used to depict results of series of experimental measurements or simulations. If result of a single measurement/simulation yields a numerical value, results of many experiments can be represented by counting numbers of experiments giving a certain value. This allows one to see the frequency of different results.

Histograms are usually fitted with a probability distribution. Typically, the plot of this distribution (See the section about function plots) occupies the same graphical space as the histogram. Figures A.5 and Figure A.5 show examples of histograms.

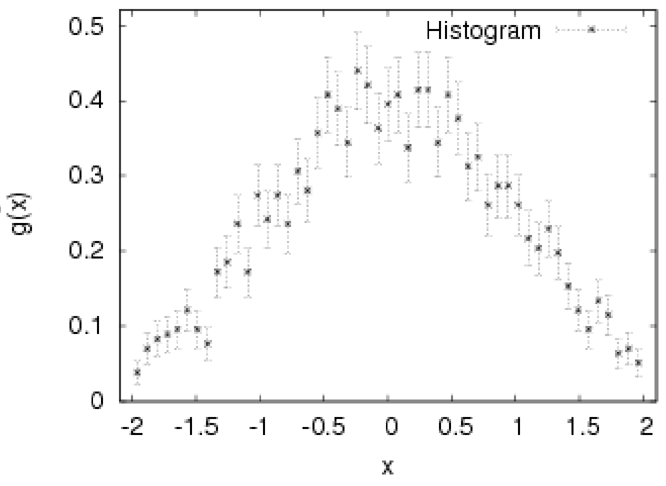


Figure A.5: Examples of histograms coming from publications stored at arXiv.org

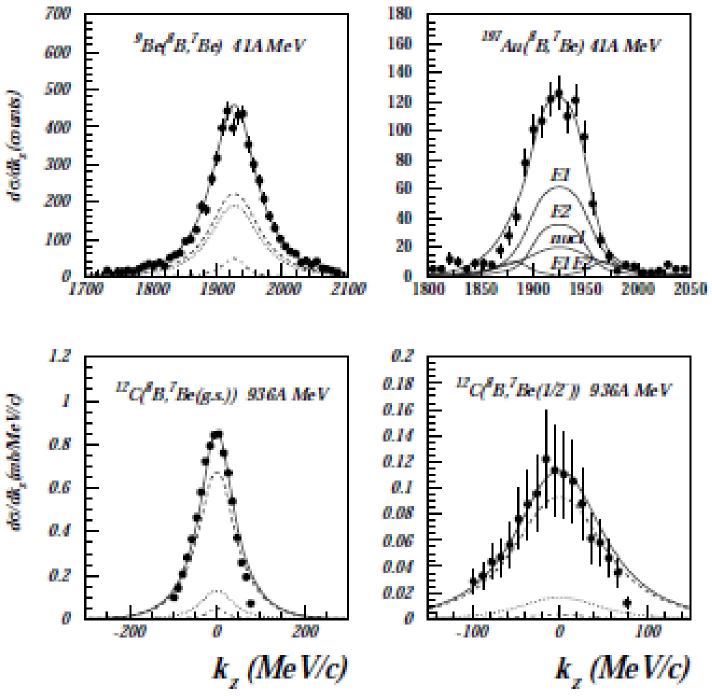


Figure A.6: Examples of histograms coming from publications stored at arXiv.org (continuation)

A.1.4 Heat Map

Heat maps are a special case of 3-dimensional plots. Instead of representing points in a 3-dimensional space using projections to multiple axes (which is ambiguous in the case of 2-dimensional representations on the graphical canvas of a figure), the value of the parameter is represented as the position in the canvas (using a similar technique for mapping as in the case of regular 2-dimensional plots). The third dimension is encoded using colours of the points.

Figure A.7 shows an example heat map.

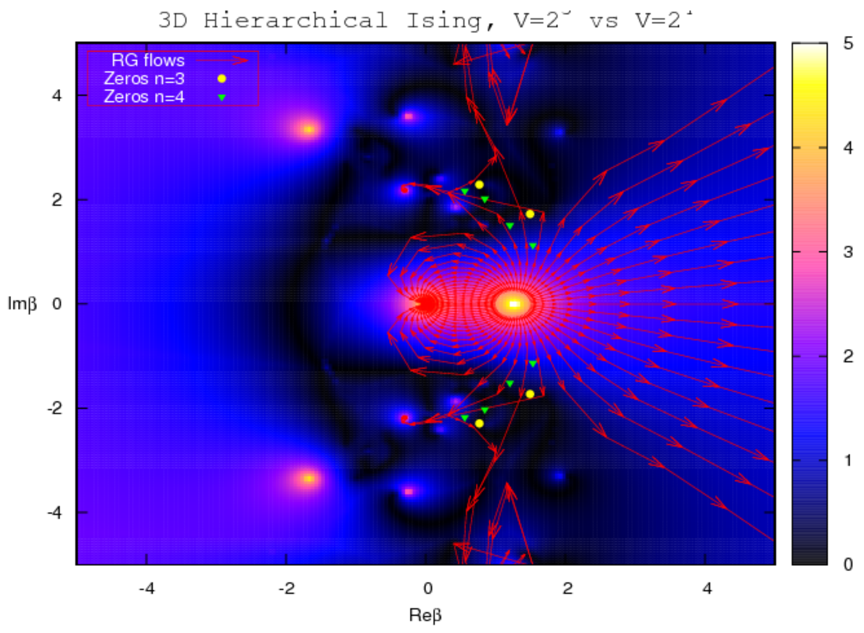


Figure A.7: An example of the heat map plot.

A.2 Diagrams

The role of diagrams is completely different from the role of plots. Instead of showing the quantitative information using the representation space, they attempt to represent the structure of abstract concepts. Figures of the ontology presented in Chapter 2 are examples of diagrams. There are types of diagrams which appear in the physics publications more frequently than the others.

A.2.1 Feynman Diagram

Feynman Diagrams are used to depict how different types of particles are transformed in time. They depict processes by showing the particles existing in the system at the initial moment and depicting all the transformations and intermediate states. Typically, edges of these diagrams represent particles and vertices represent interactions or transformations. Figure A.8 shows examples of Feynman diagrams appearing in the scholarly publications.

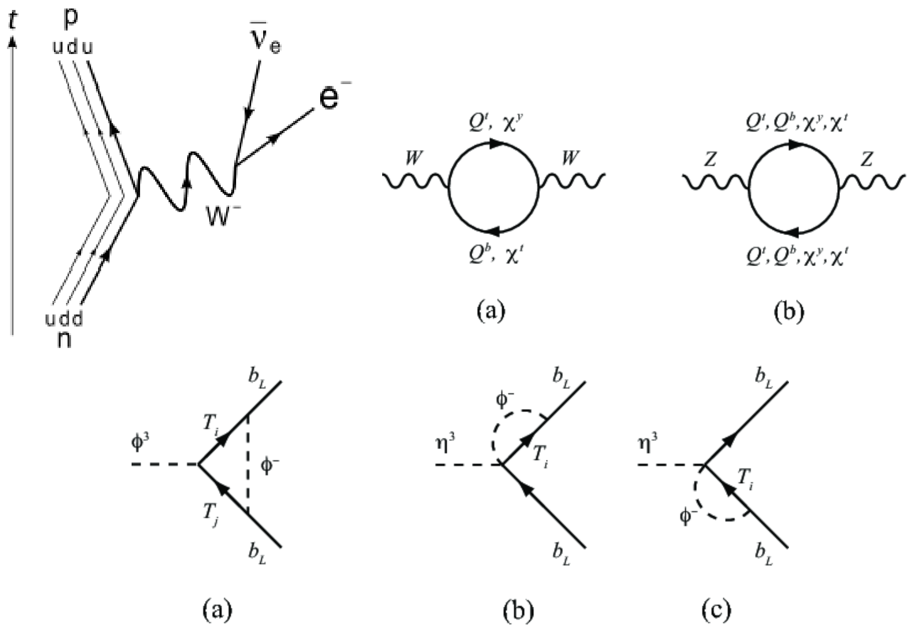


Figure A.8: Examples of Feynman diagrams coming from publications stored at arXiv.org

A.2.2 Mathematical Diagram

The role of diagrams of this type is very similar to mathematical formulas, although they are much more graphical in their nature and tend to display abstract relations between notions rather than exact dependencies. Usually, mathematical diagrams do not contain caption and rely on the description from the text of the paragraph preceding the graphics. Figure A.9 shows some of the mathematical diagrams which can be found in the literature.

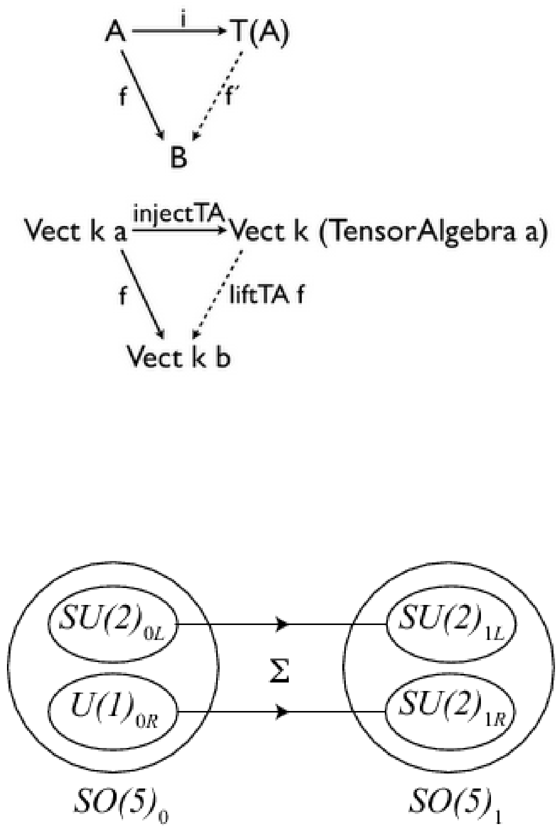


Figure A.9: Examples of mathematical diagrams coming from publications stored at arXiv.org

A.3 Pictures and schematic representations

In some cases, especially in the experimental physics, scholarly publications contain photographs of the equipment used to conduct an experiment or other types of graphics. This category includes also schematic representations of experimental devices and different custom graphics. An example can be seen in Figure A.10

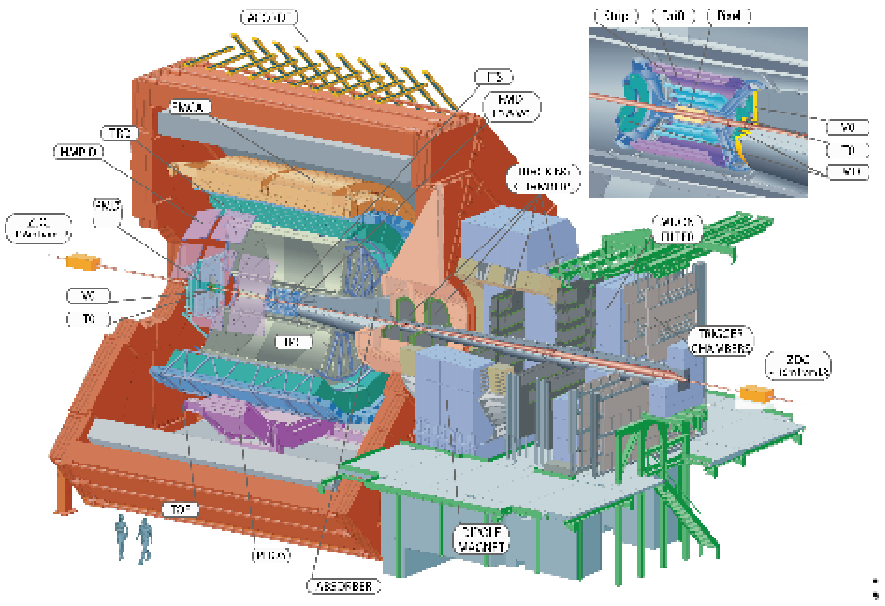


Figure A.10: Examples of custom graphics included in a publication coming from arXiv.org

Appendix B

Extensions of the BibUpload Module

This appendix describes the modifications of the *BibUpload* module, which allows manipulating the data in the new document data model. The input of a *BibUpload* process consists of a document described in a format being an extension of MARC XML. Besides correct MARC tags, this document can contain special tags which are not directly uploaded into the database but rather interpreted by the uploader. The special tags include *FFT* (standing for Fulltext File Transfer), which allows manipulating documents and FMT for updating output formats. The modifications of the data model required the extension of this format. We have extended the semantics of the *FFT* tag and added two new special tags: *BDM* (BibDoc MoreInfo) for manipulating arbitrary *MoreInfo* data; and *BDR* (BibDoc Relation) for manipulating relations between BibDocs.

Table B.1 presents the complete list of possible subfields of the *FFT* field. A detailed description of the upload process can be found in the documentation of Invenio[1]. The most important extensions to the *FFT* field consist of the introduction of *w*, *p*, *b* and *u* subfields. These subfields allow modifying the *MoreInfo* objects associated respectively with the document, a particular version of the document, a particular format of a particular version of a document and a particular format of a document regardless the version. The format and the semantics of all the subfields manipulating *MoreInfo* structures are described in the following sections.

[1]The most recent is available in the Invenio demo installation: `http://invenio-demo-next.cern.ch/help/admin/bibupload-admin-guide`

Table B.1: Possible Subfields of the FFT MARC field.

code	**description**
$a	location of the file to upload (a filesystem path or a URL)
$d	file description (optional)
$f	format (optional; if not set, deduced from $a)
$m	new desired document name (optional; used for renaming files)
$n	document name (optional; if not set, deduced from $a)
$o	flag (repeatable subfield)
$r	restriction (optional, see below)
$s	set timestamp (optional, see below)
$t	document type (e.g. Main, Additional)
$v	version (used only with REVERT and DELETE-FILE, see below)
$x	url/path for an icon (optional)
$z	comment (optional)
$w	*MoreInfo* modification of the document
$p	*MoreInfo* modification of a current version of the document
$b	*MoreInfo* modification of a current version and format of the document
$u	*MoreInfo* modification of a format (of any version) of the document

B.1 Uploading of Relations Between Documents

Uploading the relations between documents can be achieved by the usage of the *BDR* field. Table B.2 shows the complete specification of this field. A relation consists of identifiers of the source and target document with their versions and formats together with the string-valued type of the relation and optionally a *MoreInfo* specification. Not providing a field means that the relation binds documents with all possible values of a given field.

Deleting relations can be done using a special *d* subfield which should contain the text "DELETE".

B.2 Manipulations on the Custom Metadata

Extensions to MARC XML are not very appropriate for encoding the modifications to *MoreInfo* dictionaries. However, we used this approach for the simplicity and because large modifications to the Invenio data ingestion method were not encouraged. The *BibUpload* input format is only an intermediate step in the record ingestion process and the file in the input format is not stored or displayed after it served its role. This is why even a not entirely elegant solution is not very harmful at this place. However, more work could be done to provide a nicer custom metadata uploading interface.

Every MARC XML subfield intended to modify the *MoreInfo* data (regardless, in which MARC XML field it appears) has a similar structure. The content of this file encodes a Python dictionary of dictionaries. The keys of the first level represent names of the namespaces. Values

Table B.2: Possible Subfields of the BDR MARC field allowing the manipulation of the relations between the stored documents.

code	description
$r	Identifier of the relation (optional, can be provided if modifying a known relation)
$i	Identifier of the first document
$n	Name of the first document (within the current record) (optional)
$v	Version of the first document (optional)
$f	Format of the first document (optional)
$j	Identifier of the second document
$o	Name of the second document (within the current record) (optional)
$w	Version of the second document (optional)
$g	Format of the second document (optional)
$t	Type of the relation
$m	Modification of the *MoreInfo* of the relation
$d	Special field. if value=DELETE, relation is removed

in the top-level dictionary are dictionaries of real metadata. Placing a Python data structure inside a MARC field was made possible by first using the cPickle serialisation of the complete top-level dictionary object and then, in order to avoid problems with non-standard symbols, encoding the result in base64 format.

The *MoreInfo* object which should be modified is indicated by the MARC XML field and subfield, while the modifications are encoded in the aforementioned format. *BibUpload* reads and decodes the dictionary and extends the appropriate object with the passed values by overriding or adding the present keys. If a key in an internal dictionary corresponds to the None value, the key is removed from the *MoreInfo* structure.

Performing the modification of *MoreInfo* structures together with the described objects (for example as a part of the Fulltext File Transfer (FFT) or Bibdoc Relation (BDR) fields) can trigger an automatic update of the entire described object (For example a creation of a completely new version of the document). In some cases this is not desired. Invenio provides a special MARC XML field named BDM and dedicated to modify *MoreInfo* objects attached to different entities. Table B.3 provides an overview of this field.

In *BibUpload* insert and append modes cause the addition of the specified keys and namespaces. The correct and replace modes first remove the entire content of the *MoreInfo*.

Table B.3: Possible Subfields of the BDM MARC field that allow manipulating *MoreInfo*.

code	description
$r	Identifier of a relation between documents (optional)
$i	Identifier of a BibDoc (optional)
$v	Version of a BibDoc (optional)
$n	Name of a BibDoc (within a current record) (optional)
$f	Format of a BibDoc (optional)
$m	Serialised update to the *MoreInfo* dictionary

B.3 Interlinking Between Newly Created Objects

All the persistent identifiers like the number of a record or a number of a document are generated during the execution of *BibUpload*. However, certain places inside the input MARC XML require these identifiers to be specified. This leads to difficulties when trying to provide an identifier which has not yet been generated. Such a situation can arise for example when trying to upload two documents and a relation between them using the same *BibUpload* process.

Our extensions of the document storage model provide a solution for such situations. Now, *BibUpload* provides a substitute for the persistent identifiers. In every place which requires the usage of an identifier or the number of a version of a document, the user can now specify the temporary identifiers. During the processing by the *BibUpload* process, these identifiers are replaced with newly created persistent identifiers.

A temporary identifier can be an arbitrary name, which is unique in the scope of the entire *BibUpload* input file. It differs from a persistent identifier because its name is prefixed by the "TMP:" string. There are two separate namespaces for the temporary identifiers: the one for versions, and the one for identifiers of the documents. This prevents users from specifying a version number as the identifier of a document and vice versa. It also means that "TMP:id" used in the context of an identifier of a document and the same temporary identifier used in the context of the version of the same document are interpreted as two separate temporary identifiers.

Appendix C

The Training Procedure for the Coordinate System Detection SVM

This appendix describes the technical aspects of the procedure which we have used to train the Support Vector Machine (SVM) classifier detecting coordinate systems in the semantics extraction procedure described in Chapter 4. The extractor of semantics analyses an extracted figure to detect a number of possible coordinate systems. Not all of the detected coordinate system candidates correspond to the real coordinate systems inside a figure. In the method described in Chapter 4, we propose the usage of SVM to distinguish between the coordinate systems and the incorrectly detected candidates. The usage of SVMs requires the samples to be represented as vectors in a certain vector space. Vectorisation of relatively abstract entities as coordinate systems can be achieved in many different ways. Those different vectorisations together with different choices of SVM parameters can lead to a different accuracy of the classification. Experimenting with different vectorisations led us to design a training procedure which minimises the effort required to redesign the usage of SVMs. In this appendix we explain and justify the procedure of training SVM classifiers for coordinate system detection.

C.1 The Overview of the Training Process

The procedure of detecting coordinate system candidates has been designed in a manner which minimises the chance of omitting real coordinate system. As a consequence, the number of coordinate system candidates detected in separate figures is high. Processing a random sample of 509 automatically extracted figures resulted in 63106 figure candidates. This corresponds to the average of 124 coordinate system candidates per figure. At the same time, the number of real coordinate systems present in every figure is low. Typically, only plots contain coordinate systems and only part of all the figures consists of plots. A typical plot contains 1-2 coordinate systems. In the extreme cases, the number of the coordinate systems rarely exceeds 10.

Coordinate system candidates extracted from a single plot tend to be similar. As such, their representations in a vector space used for the SVM classification tend to be close to each other. The effectiveness of the SVM training depends on how representative is the used set of samples. Using a large number of similar figure candidates does not guarantee a high efficiency in solving the problem of classification. The variety of samples used for the classification can be assured by extracting them from a larger number of figures.

After the extraction, the training samples need to be manually classified as belonging to one of two categories: coordinate systems and false candidates. The manual classification of a large number of samples is a laborious process. We designed the SVM-training procedure in a manner which minimises the volume of the required manual work and yet makes the samples less concentrated in particular points of the feature space. Instead of using a small number of figures and classifying all the detected coordinate system candidates, we have decided to randomly select the candidates which should be further manually classified. Out of 63106 extracted coordinates, we chose 4000. Their manual classification has unveiled 257 true coordinate system candidates. The number of real coordinate systems is low comparing to the number of the detected coordinate system candidates. This is also true after the random selection of samples. In order to assure a wider representation of the real coordinate system candidates, we have enriched the randomly selected sample set with a number of correct coordinate systems.

We experimented with different vectorisations of coordinate system candidates. The training procedures had to allow simple reevaluation of different representations as elements of a vector space without repeating the manual classification work. Figure C.1 summarises the entire SVM-related workflow. First, the selected figures are processed and samples are generated. Every coordinate system candidate is identified in a manner that allows its unique identification during the subsequent executions of the extraction. The generated samples are subject to the manual classification. This procedure has to be executed only once every time the training set has been changed, which might happen when for example the extractor is to be used with a different corpus of figures. The training of the classifier begins with the vectorisation of the classified candidates and the construction of the training set. The training set is passed to the SVM library and a trained model is produced. This model is saved in an external file and used during the classification of the samples.

C.2 The Manual Classification

Manually classifying the samples would be a difficult task if only the abstract numeric representations of the coordinate system candidates were available. Instead, the procedure of preparing samples for classification, creates a graphical file for every of the extracted candidates. The

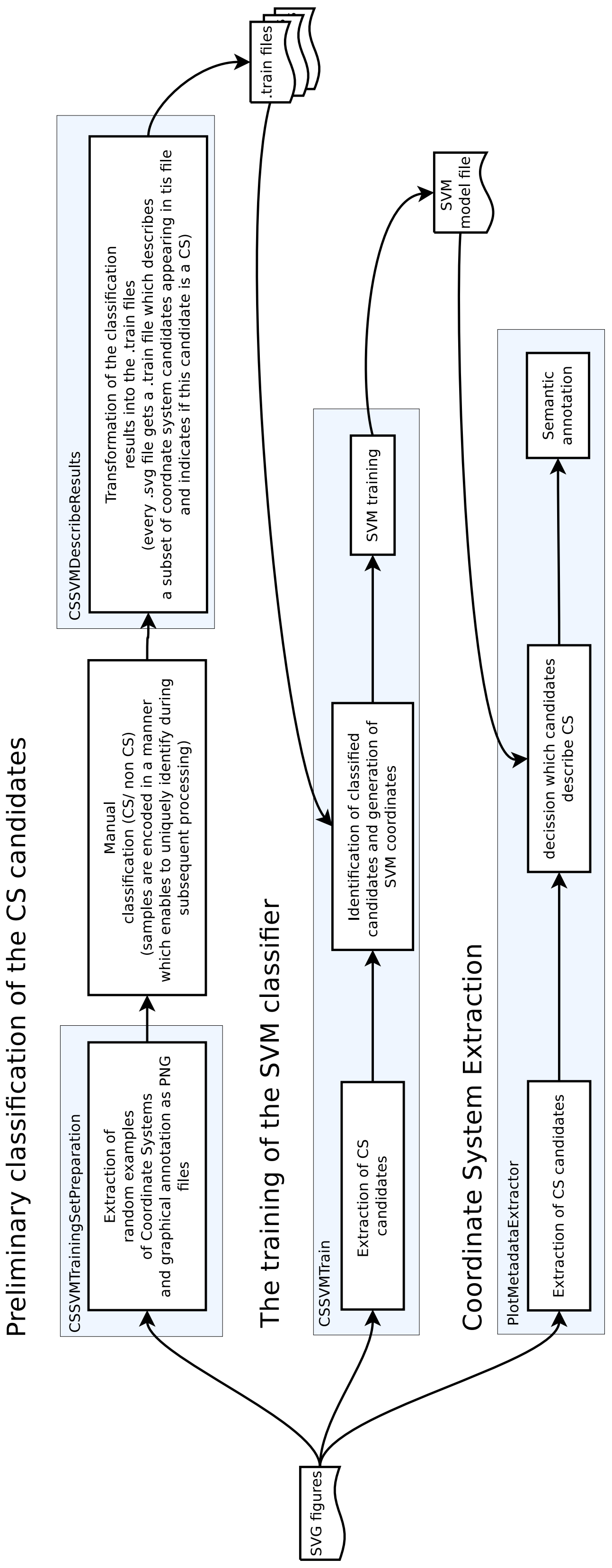
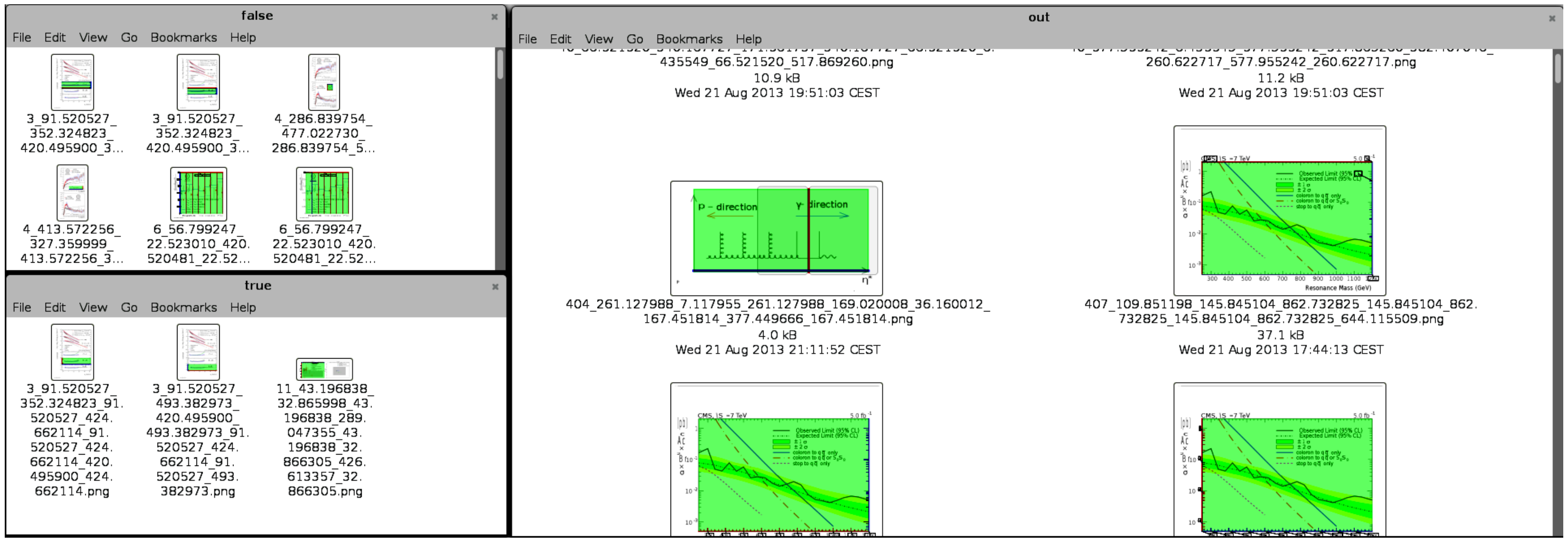

Preliminary classification of the CS candidates
CSSVMTrainingSetPreparation
Extraction of random examples of Coordinate Systems and graphical annotation as PNG files
Manual classification (CS/ non CS) (samples are encoded in a manner which enables to uniquely identify during subsequent processing)
CSSVMDescribeResults
Transformation of the classification results into the .train files (every .svg file gets a .train file which describes a subset of coordnate system candidates appearing in tis file and indicates if this candidate is a CS)
.train files
The training of the SVM classifier
CSSVMTrain
Extraction of CS candidates
Identification of classified candidates and generation of SVM coordinates
SVM training
SVM model file
Coordinate System Extraction
PlotMetadataExtractor
Extraction of CS candidates
decission which candidates describe CS
Semantic annotation
SVG figures

Figure C.1: Overview of the different processes used to train SVM


false
File Edit View Go Bookmarks Help
true
File Edit View Go Bookmarks Help
out
File Edit View Go Bookmarks Help

3_91.520527_352.324823_420.495900_3...
3_91.520527_352.324823_420.495900_3...
4_286.839754_477.022730_286.839754_5...
4_413.572256_327.359999_413.572256_3...
6_56.799247_22.523010_420.520481_22.52...
6_56.799247_22.523010_420.520481_22.52...

3_91.520527_352.324823_91.520527_424.662114_91.520527_424.662114_420.495900_424.662114.png
3_91.520527_493.382973_420.495900_493.382973_91.520527_424.662114_91.520527_493.382973.png
11_43.196838_32.865998_43.196838_289.047355_43.196838_32.866305_426.613357_32.866305.png

435549_66.521520_517.869260.png
10.9 kB
Wed 21 Aug 2013 19:51:03 CEST

260.622717_577.955242_260.622717.png
11.2 kB
Wed 21 Aug 2013 19:51:03 CEST

404_261.127988_7.117955_261.127988_169.020008_36.160012_167.451814_377.449666_167.451814.png
4.0 kB
Wed 21 Aug 2013 21:11:52 CEST

407_109.851198_145.845104_862.732825_145.845104_862.732825_145.845104_862.732825_644.115509.png
37.1 kB
Wed 21 Aug 2013 17:44:13 CEST

Figure C.2: The manual classification of samples

file depicts the entire figure which is subject to the extraction procedure. Axes of a coordinate system candidate are represented with bold lines and ticks with regular width lines. The labels matched with axis ticks are marked with rectangles and connected with their corresponding axis tick. Additionally, all the area spanned by the coordinate system is highlighted with a green rectangle. The name of the file encodes all the numerical parameters of the CS candidate, which are necessary for the algorithm to distinguish this candidate from others. This information consists of the name of the source file and the coordinates of axis lines. As the result, the content of the graphical files is easily interpretable by human users. The name of the file is machine-readable.

The classification of the prepared samples can be done by moving the extracted files into one of two directories ("true" for real coordinate systems and "false" for the false candidates). This task can be facilitated by one of the file managers which previsualise miniatures of the images. Figure C.2 shows the *Nautilus* file manager displaying the miniatures of the described coordinate system candidates. As it can be seen in the picture, in many cases the displayed miniature is sufficient to classify a sample as true or false candidate.

The "false" and "true" folders resulting from the manual classification are further subject to the automatic interpretation. The algorithm reads both folders and generates a description of .train files. Those files contain only machine-readable entries together with the information if a given sample is true or false candidate.

Appendix D

Acronyms

AI Artificial Intelligence 55

AJAX Asynchronous Asynchronous Javascript and XML

API Application Programming Interface

ASCII American Standard Code for Information Interchange

BDM Bibdoc MoreInfo

BDR Bibdoc Relation 153

CDS CERN Document Server 88

CERN European Organisation for Nuclear Research 2

CREAM Cosmic Ray Energetics And Mass 77

CSS Cascade Style Sheets 113

CV	Computer Vision	55
DESY	Deutsches Elektronen-Synchrotron	3
DL	Description Logic	8
DOI	Digital Object Identifier	
FFT	Fulltext File Transfer	153
HEP	High Energy Physics	2
HFO	Hep Figures Ontology	8
HTML	HyperText Markup Language	
HTTP	HyperText Transfer Protocol	133
IC	Information Content	84
IR	Information Retrieval	53
IT	Information Technology	
JPEG	Joint Photographic Experts Group	
kmph	kilometres per hour	19
LHEF	Les Houches Event File	11

MARC	MAchine-Readable Cataloging	88
MC	Monte Carlo	9
MUO	Measurable Units Ontology	19
NLP	Natural Language Processing	71
OAI	Open Archives Initiative	
ODT	Open Document Text	
OWL	Web Ontology Language	8
PCA	Primary Component Analysis	56
PDF	Portable Document Format	27
PDG	Particle Data Group	8
PMH	Protocol for Metadata Harvesting	
PNG	Portable Network Graphics	43
PS	PostScript	
RDF	Resource Description Framework	8
SIO	Semantic Science Integrated Ontology	11

SKOS	Simple Knowledge Organization System	8
SLAC	Stanford Linear Accelerator Center	2
SPARQL	SPARQL Protocol and RDF Query Language	67
SPIRES	Stanford Public Information Retrieval System	2
SQL	Structured Query Language	
SVD	Singular Value Decomposition	56
SVG	Scalable Vector Graphics	43
SVM	Support Vector Machines	52
TBox	Terminological Box	56
URI	Unified Resource Identifier	102
URL	Uniform Resource Locator	
WWW	World Wide Web	2
XML	Extensible Markup Language	8

Bibliography

[1] Abadi, D. J., Marcus, A., Madden, S. R., and Hollenbach, K. Scalable semantic web data management using vertical partitioning. In *Proceedings of the 33rd international conference on Very large data bases* (2007), VLDB '07, VLDB Endowment, pp. 411–422.

[2] Alwall, J., Ballestrero, A., Bartalini, P., Belov, S., Boos, E., et al. A Standard format for Les Houches event files. *Comput.Phys.Commun. 176* (2007), 300–304.

[3] Antoniou, G., and vanHarmelen, F. *A Semantic Web Primer.* MIT Press, Cambridge, MA, USA, 2004.

[4] Baader, F., Calvanese, D., McGuinness, D. L., Nardi, D., and Patel-Schneider, P. F., Eds. *The description logic handbook: theory, implementation, and applications.* Cambridge University Press, New York, NY, USA, 2003.

[5] Baeza-Yates, R., and Ribeiro-Neto, B. *Modern information retrieval.* ACM Press Books. Addison-Wesley, Reading, MA, 1999.

[6] Ballard, D. H. Readings in computer vision: issues, problems, principles, and paradigms. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1987, ch. Generalizing the hough transform to detect arbitrary shapes, pp. 714–725.

[7] Belov, S., Dudko, L., Kekelidze, D., and Sherstnev, A. HepML, an XML-based format for describing simulated data in high energy physic. *Comput.Phys.Commun. 181* (2010), 1758–1768.

[8] Bhatia, S., Lahiri, S., and Mitra, P. Generating synopses for document-element search. In *Proceeding of the 18th ACM conference on Information and knowledge management* (New York, NY, USA, 2009), CIKM '09, ACM, pp. 2003–2006.

[9] Browuer, W., Kataria, S., Das, S., Mitra, P., and Giles, C. L. Segregating and extracting overlapping data points in two-dimensional plots. In *Proceedings of the 8th ACM/IEEE-CS joint conference on Digital libraries* (New York, NY, USA, 2008), JCDL '08, ACM, pp. 276–279.

[10] Caffaro, J., and Kaplun, S. Invenio: A Modern Digital Library for Grey Literature. oai:cds.cern.ch:1312678. In *12th. Int. Conf. on Grey Literature, Prague, Czech Republic* (Geneva, Dec 2010), no. CERN-OPEN-2010-027, p. 7.

[11] Chao, H., and Fan, J. Layout and content extraction for pdf documents. In *Document Analysis Systems* (2004), pp. 213–224.

[12] Cleveland, W. S. Graphs in Scientific Publications. *The American Statistician 38*, 4 (1984), 261–269.

[13] Cormen, T. H., Leiserson, C. E., and Rivest, R. L. *Introduction to Algorithms.* MIT electrical engineering and computer science series. MIT, Cambridge, 1990.

[14] Cortes, C., and Vapnik, V. Support-vector networks. In *Machine Learning* (1995), pp. 273–297.

[15] Dasiopoulou, S., Kompatsiaris, I., and Strintzis, M. G. Applying fuzzy dls in the extraction of image semantics. *J. Data Semantics 14* (2009), 105–132.

[16] Daubechies, I. The wavelet transform, time-frequency localization and signal analysis. *IEEE Trans. Inf. Theor. 36*, 5 (Sept. 2006), 961–1005.

[17] de Sompel, H. V., Nelson, M. L., Lagoze, C., and Warner, S. Resource Harvesting within the OAI-PMH Framework. *D-Lib Magazine 10*, 12 (2004).

[18] Duda, R. O., and Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. *Commun. ACM 15*, 1 (Jan. 1972), 11–15.

[19] Dumontier, M., Ferres, L., and Villanueva-Rosales, N. Modeling and querying graphical representations of statistical data. *Web Semantics: Science, Services and Agents on the World Wide Web 8*, 2-3 (2010), 241–254.

[20] Edelsbrunner, H., and Maurer, H. A. On the intersection of orthogonal objects. *Information Processing Letters 13* (1981).

[21] Eichhorn, G. Trends in Scientific Publishing at Springer. In *Future Professional Communication in Astronomy II* (2011), Astrophysics and Space Science Proceedings, Springer.

[22] Elsevier. SciVerse Science Direct: Image Search. `http://www.info.sciverse.com/sciencedirect/using/searching-linking/image`, 2012. last access: 6 November 2012.

[23] Ferraiolo, J., Ed. *Scalable Vector Graphics (SVG) 1.0 Specification.* Iuniverse Inc, 2001.

[24] Gándara, A., and Villanueva-Rosales, N. Documenting and sharing scientific research over the semantic web. In *Proceedings of the 12th International Conference on Knowledge Management and Knowledge Technologies* (New York, NY, USA, 2012), i-KNOW '12, ACM, pp. 18:1–18:8.

[25] Gentil-Beccot, A., Mele, S., Holtkamp, A., O'Connell, H. B., and Brooks, T. C. Information Resources in High-Energy Physics: Surveying the Present Landscape and Charting the Future Course. oai:cds.cern.ch:1099955. *J. Am. Soc. Inf. Sci. Technol. 60*, arXiv:0804.2701. CERN-OPEN-2008-010. DESY-08-040. DESY-2008-040. FERMILAB-PUB-08-077-BSS. SLAC-PUB-13199. 1 (Apr 2008), 150–160. 27 p.

[26] Hearst, M. A., Divoli, A., Ye, J., and Wooldridge, M. A. Exploring the efficacy of caption search for bioscience journal search interfaces. In *Proceedings of the Workshop on BioNLP 2007: Biological, Translational, and Clinical Language Processing* (2007), BioNLP '07, pp. 73–80.

[27] Holzner, A. G., Igo-Kemenes, P., and Mele, S. First results from the PARSE.Insight project: HEP survey on data preservation, re-use and (open) access. In *Proceedings of First Workshop on Data Preservation and Long-Term Analysis in High-Energy Physics* (DESY, Hamburg, Germany, January 26-28 2009).

[28] Huang, J., Abadi, D. J., and Ren, K. Scalable sparql querying of large rdf graphs. *PVLDB 4*, 11 (2011), 1123–1134.

[29] Johnston, L. Web Reviews: See the Science: SciTech Image Databases. *Sci-Tech News 65* (2011).

[30] Kataria, S. On utilization of information extracted from graph images in digital documents. *Bulletin of IEEE Technical Comittee on Digital Libraries 4* (2008).

[31] Kataria, S., Browuer, W., Mitra, P., and Giles, C. L. Automatic extraction of data points and text blocks from 2-dimensional plots in digital documents. In *Proceedings of the 23rd national conference on Artificial intelligence - Volume 2* (2008), AAAI Press, pp. 1169–1174.

[32] Kittur, A., Chi, E. H., and Suh, B. Crowdsourcing user studies with mechanical turk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (New York, NY, USA, 2008), CHI '08, ACM, pp. 453–456.

[33] Lacasta, J., Nogueras-Iso, J., Teller, J., and Falquet, G. Transformation of a keyword indexed collection into a semantic repository: applicability to the urban domain. In *Proceedings of the 15th international conference on Theory and practice of digital libraries: research and advanced technology for digital libraries* (Berlin, Heidelberg, 2011), TPDL'11, Springer-Verlag, pp. 372–383.

[34] Liu, Y., Bai, K., Mitra, P., and Giles, C. L. TableSeer: Automatic Table Metadata extraction and Searching in Digital Libraries. In *JCDL'07, June 18-23, 2007* (Vancouver, British Columbia, Canada, 2007), JCDL.

[35] Lu, X., Wang, J. Z., Mitra, P., and Giles, C. L. Automatic extraction of data from 2-d plots in documents. In *INTERNATIONAL CONFERENCE ON DOCUMENT ANALYSIS AND RECOGNITION* (2007), IEEE Computer Society, pp. 188–192.

[36] Mallat, S. G. A theory for multiresolution signal decomposition: the wavelet representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence 11* (1989), 674–693.

[37] Niknam, M., and Kemke, C. Modeling Shapes and Graphics Concepts in an Ontology. *Proceedings of the First International Workshop on SHAPES, Karlsruhe 812* (2011).

[38] Pearson, K. On lines and planes of closest fit to systems of points in space. *Philosophical Magazine 2*, 6 (1901), 559–572.

[39] Pepe, A., and Yeomans, J. Protocols for scholarly communication. In *5th Conference on Library and Information Services in Astronomy* (Cambridge, MA, USA, 2006).

[40] Pesquita, C., Faria, D., Falcao, A. O., Lord, P., and Couto, F. M. Semantic similarity in biomedical ontologies. *PLoS Comput Biol. 5* (2009).

[41] Praczyk, P. Apache Hadoop - Wolna implementacja Map Reduce (Apache Hadoop - The Free Implementation of Map Reduce). In *Jesien Linuksowa (The Polish free software and GNU/Linux conference)* (Huta Szklana, 2-4 October 2009 (`http://jesien.linux.org.pl/2009/ksiunszka/jesienbook-2009.pdf`)), pp. 58–65.

[42] Praczyk, P. A., and Nogueras-Iso, J. A semantic approach for the annotation of figures in High-Energy Physics. In *Accepted for publication in Proc. of MTSR 2013 - 7th Metadata and Semantics Research Conference, Thesaloniki, Greece, November 19-22, 2013. Springer, vol. 390 of Communications in Computer and Information Science (CCIS), 12 pages* (2013).

[43] Praczyk, P. A., and Nogueras-Iso, J. Automatic Extraction of Figures from Scientific Publications in High-Energy Physics. *To appear in: Information Technology and Libraries, ISSN 0730-9295, 21 pages.* (2013).

[44] Praczyk, P. A., Nogueras-Iso, J., Dallemeier-Tiessen, S., and Whalley, M. Integrating Scholarly Publications and Research Data - Preparing for Open Science, a case Study from High-Energy Physics with Special Emphasis on (Meta)data Models. In *Metadata and Semantics Research* (`http://link.springer.com/content/pdf/10.1007%2F978-3-642-35233-1_16`, 2012), vol. 343 of *Communications in Computer and Information Science (CCIS)*, Springer, pp. 146–157.

[45] Praczyk, P. A., Nogueras-Iso, J., Kaplun, S., and Simko, T. A Storage Model for Supporting Figures and Other Artefacts in Scientific Libraries: the Case Study of Invenio.

In *Proc. of 4th Workshop on Very Large Digital Libraries (VLDL 2011)* (Berlin, Germany, (Workshop in conjunction with the 1st International Conference on Theory and Practice of Digital Libraries (TPDL)), 13 pages., 2011).

[46] Rowley, H. A., Baluja, S., and Kanade, T. Neural network-based face detection. *IEEE Transactions On Pattern Analysis and Machine intelligence 20* (1998), 23–38.

[47] Russell, S., and Norvig, P. *Artificial Intelligence: A Modern Approach*, 3 ed. Prentice Hall, Dec. 2009.

[48] Samadian, S., McManus, B., and Wilkinson, M. Extending and encoding existing biological terminologies and datasets for use in the reasoned semantic web. *J Biomed Semantics 3*, 1 (2012), 6.

[49] Shannon, C. E., Weaver, W., and Shannon. *The Mathematical Theory of Communication.* University of Illinois Press, Sept. 1998.

[50] Theodoridis, S., and Koutroumbas, K. *Pattern Recognition, Third Edition.* Academic Press, February 2006.

[51] Treil, N., Mallat, S. G., Bajcsy, R., and States., U. *Image wavelet decomposition and applications [microform] / N. Treil, S. Mallat, R. Bajcsy.* Dept. of Computer and Information Science, School of Engineering and Applied Science, University of Pennsylvania ; National Aeronautics and Space Administration ; National Technical Information Service, distributor Philadelphia, PA : [Washington, DC : Springfield, Va], 1989.

[52] Vesely, M., Baron, T., Le Meur, J.-Y., and Simko, T. Creating Open Digital Library Using XML: Implementation of OAi-PMH Protocol at CERN. oai:cds.cern.ch:590906. In *Int. Conf. on Electronic Publishing, Karlovy Vary, Czech Republic* (Jul 2002), no. CERN-ETT-2002-003, p. 7.

Index