

Tutorial

LNBIP 138

Marie-Aude Aufaure
Esteban Zimányi (Eds.)

Business Intelligence

Second European Summer School, eBISS 2012
Brussels, Belgium, July 2012
Tutorial Lectures

 Springer

Lecture Notes in Business Information Processing

138

Series Editors

Wil van der Aalst

Eindhoven Technical University, The Netherlands

John Mylopoulos

University of Trento, Italy

Michael Rosemann

Queensland University of Technology, Brisbane, Qld, Australia

Michael J. Shaw

University of Illinois, Urbana-Champaign, IL, USA

Clemens Szyperski

Microsoft Research, Redmond, WA, USA

Marie-Aude Aufaure
Esteban Zimányi (Eds.)

Business Intelligence

Second European Summer School, eBISS 2012
Brussels, Belgium, July 15-21, 2012
Tutorial Lectures



Springer

Volume Editors

Marie-Aude Aufaure
Ecole Centrale Paris
MAS Laboratory
Châtenay-Malabry, France
E-mail: marie-aude.aufaure@ecp.fr

Esteban Zimányi
Université Libre de Bruxelles
Department of Computer and Decision Engineering (CoDE)
Brussels, Belgium
E-mail: ezimanyi@ulb.ac.be

ISSN 1865-1348 e-ISSN 1865-1356
ISBN 978-3-642-36317-7 e-ISBN 978-3-642-36318-4
DOI 10.1007/978-3-642-36318-4
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012956061

ACM Computing Classification (1998): J.1, H.2, H.3, D.1, G.3, G.2

© Springer-Verlag Berlin Heidelberg 2013

This work is subject to copyright. All rights are reserved, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, re-use of illustrations, recitation, broadcasting, reproduction on microfilms or in any other way, and storage in data banks. Duplication of this publication or parts thereof is permitted only under the provisions of the German Copyright Law of September 9, 1965, in its current version, and permission for use must always be obtained from Springer. Violations are liable to prosecution under the German Copyright Law.

The use of general descriptive names, registered names, trademarks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

Typesetting: Camera-ready by author, data conversion by Scientific Publishing Services, Chennai, India

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Preface

The Second European Business Intelligence Summer School (eBISS 2012) took place in Brussels, Belgium, in July 2012. Tutorials were given by experts in business intelligence and covered various hot topics. This volume contains the lecture notes of the summer school.

The first chapter shows how to support multidimensional business intelligence for complex application domains such as medical data, location-based services, music data, web data, and text data. In particular, complex dimension hierarchies, complex measures, and integration of multidimensional data with complex external data are detailed.

The second chapter introduces business process modeling. Business process management can be seen as an extension of classical workflow management systems. This chapter discusses the similarities and differences between these two notions, and then introduces the current OMG standard BPMN 2.0 (Business Process Modeling and Notation) and discusses BPEL (Business Process Execution Language). Finally, hot research topics in this field such as business process mining are depicted.

The third chapter provides an overview of machine learning techniques in time series forecasting, and outlines the challenging issues related to predicting the future in applied sciences. Three aspects are discussed in this chapter: the formalization of one-step forecasting problems as supervised learning tasks, the discussion of local learning techniques as an effective tool for dealing with temporal data, and the role of the forecasting strategy when we move from one-step to multiple-step forecasting.

The fourth chapter gives an overview of Markov logic networks from a theoretical and a practical viewpoint. Statistical relational learning approaches, more specifically Bayesian logic networks, are then detailed. Inferencing and learning processes are then explained together with the best scaling algorithms known today. This chapter concludes with an overview of application areas.

The fifth chapter focuses on recent developments, challenges, and potential solutions for mining large graphs. The main challenge of new tools and frameworks lies in the development of new paradigms that are scalable, efficient, and flexible. Distributed computing is depicted through the MapReduce paradigm. Finally, a new field of research, graph data warehousing, which is deeply linked with large graph mining, is introduced.

The sixth chapter addresses the challenges induced by big data analytics on modern architectures. Massively parallel analysis systems and their programming models are discussed, as well as the application of these modern architectures on database processing.

The seventh chapter introduces decision aid, and more specifically multicriteria decision aid, with a focus on two methods: PROMETHEE and GAIA. An illustrative example, highlighting the added value of using interactive and visual tools in complex decision processes, is analyzed with the D-Sight software.

The eighth chapter explores the importance of semantic technologies (ontologies) and knowledge extraction techniques for knowledge management, search, and capture in e-business processes. Semantic technologies and ontology learning from web data are detailed, and the use of ontologies for business intelligence is discussed through use cases described in several fields.

Finally, the ninth chapter presents the Business Semantics Management (BSM) method, a fact-oriented approach to knowledge modeling grounded in natural language. This method constitutes an interface between Enterprise Information Management and the Web of Data. BSM was implemented in the Flemish Public Administration for building the Flanders Research Information Space (FRIS) program.

We would like to thank the attendants of the summer school for their active participation, as well as the speakers and their co-authors for the high quality of their contributions in a constantly evolving and highly competitive domain. Finally, the lectures in this volume greatly benefited from the comments of the external reviewers.

November 2012

Marie-Aude Aufaure
Esteban Zimányi
eBISS 2012 Co-chairs

Organization

The Second European Business Intelligence Summer School (eBISS 2012) was organized by the MAS Laboratory of the Ecole Centrale de Paris and the Department of Computer and Decision Engineering (CoDE) of the Université Libre de Bruxelles.

Program Committee

| | |
|--------------------|---|
| Alberto Abelló | Universitat Politècnica de Catalunya, Spain |
| Marie-Aude Aufaure | Ecole Centrale de Paris, France |
| Patrick Marcel | Université François Rabelais de Tours, France |
| Alexander Löser | Technische Universität Berlin, Germany |
| Esteban Zimányi | Université Libre de Bruxelles, Belgium |

Local Organizers

| | |
|---|--|
| Angélique Dufresnes | Université Libre de Bruxelles, Belgium |
| Alejandro Vaisman (Organization Chair) | Université Libre de Bruxelles, Belgium |
| Stijn Vansummeren | Université Libre de Bruxelles, Belgium |
| Boris Verhaegen | Université Libre de Bruxelles, Belgium |
| Gary Verhaegen | Université Libre de Bruxelles, Belgium |
| Esteban Zimányi | Université Libre de Bruxelles, Belgium |

External Referees

| | |
|-------------------|---|
| Mohammad Al Hasan | Indiana University-Purdue University Indianapolis, USA |
| Omar Boussaid | ERIC Laboratory, University of Lyon 2, France |
| Paola Cerchiello | University of Pavia, Italy |
| Giorgio Corani | IDSIA (Istituto Dalle Molle di Studi sull'Intelligenza Artificiale), Manno, Switzerland |
| Etienne Cuvelier | Ecole Centrale Paris, France |
| Daniel Deutch | Ben Gurion University, Israel |
| Luis Dias | University of Coimbra, Portugal |
| Marlon Dumas | University of Tartu, Estonia |

VIII Organization

Bingsheng He

Mustafa Jarrar

Gabriele Kern-Isberner

Jens Lechtenböcker

Fionn Murtagh

Wim Peters

Hans Weigand

Nanyang Technological University, Singapore

Birzeit University, Palestine

Technische Universität Dortmund, Germany

Westfälische Wilhelms-Universität, Münster,
Germany

Science Foundation, Ireland

University of Sheffield, UK

Tilburg University, The Netherlands

Table of Contents

| | |
|---|-----|
| Managing Complex Multidimensional Data | 1 |
| <i>Torben Bach Pedersen</i> | |
| An Introduction to Business Process Modeling | 29 |
| <i>Alejandro Vaisman</i> | |
| Machine Learning Strategies for Time Series Forecasting | 62 |
| <i>Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne</i> | |
| Knowledge Discovery from Constrained Relational Data: A Tutorial on Markov Logic Networks | 78 |
| <i>Marcus Spies</i> | |
| Large Graph Mining: Recent Developments, Challenges and Potential Solutions | 103 |
| <i>Sabri Skhiri and Salim Jouili</i> | |
| Big Data Analytics on Modern Hardware Architectures: A Technology Survey | 125 |
| <i>Michael Saecker and Volker Markl</i> | |
| An Introduction to Multicriteria Decision Aid: The PROMETHEE and GAIA Methods | 150 |
| <i>Yves De Smet and Karim Lidouh</i> | |
| Knowledge Harvesting for Business Intelligence | 177 |
| <i>Nesrine Ben Mustapha and Marie-Aude Aufaure</i> | |
| Business Semantics as an Interface between Enterprise Information Management and the Web of Data: A Case Study in the Flemish Public Administration | 208 |
| <i>Christophe Debruyne and Pieter De Leenheer</i> | |
| Author Index | 235 |

Managing Complex Multidimensional Data

Torben Bach Pedersen

Aalborg University, 9220 Aalborg Ø, Denmark

`tbp@cs.aau.dk`

<http://people.cs.aau.dk/~tbp>

Abstract. Multidimensional database concepts such as cubes, dimensions with hierarchies, and measures are a cornerstone of business intelligence. However, the standard data models and system implementations (OLAP) for multidimensional databases are sometimes not able to capture the complexities of advanced real-world application domains. This lecture will focus on how to manage such complex multidimensional data, including complex dimension hierarchies, complex measures, and integration of multidimensional data with complex external data. We will look at how complex multidimensional data emerge in complex application domains such as medical data, location-based services, music data, web data, and text data, and present solutions for these domains that support multidimensional business intelligence.

Keywords: multidimensional databases, dimensions, hierarchies, measures, complex data.

1 Introduction

This paper concerns the management of complex multidimensional data. Multidimensional database concepts such as cubes, dimensions with hierarchies, and measures have become essential concepts for the increasingly important area of business intelligence. A wide range of standard data models and system implementations (OLAP) for multidimensional databases have emerged, leading to a multibillion dollar industry. However, these standard models and tools are sometimes not able to capture the complexities of advanced real-world application domains such as medical data, location-based services, music data, web data, and text data. This lecture will focus on how to manage such complex multidimensional data, including complex dimension hierarchies, complex measures, and integration of multidimensional data with complex external data. The lecture complements an invited lecture by the author on his research at the eBISS 2012 summer school, and is thus primarily based on the work of the author and not aimed to provide detailed coverage of all similar work, as this is not possible within the given limits. The lecture is structured as follows: First, it will introduce related terminology and the history of multidimensional databases, followed by a discussion of why wellknown concepts such as spreadsheets and relational databases do not suffice in this setting. Second, to give readers new

to the area of multidimensional databases a foundation for the further reading, the lecture briefly introduces standard multidimensional database concepts such as data cubes, dimensions with hierarchies, facts, measures, and the associated multidimensional querying operators. Third, the lecture will introduce the area of complex multidimensional data, giving examples of the complexities that may occur. Fourth, the lecture provides a set of requirements for handling complex multidimensional data and surveys existing multidimensional models with respect to these requirements. Fifth, the lecture discusses how complex multidimensional data emerge in complex application domains, and briefly present solutions for handling the complex multidimensional data. Finally, the lecture summarizes the paper and points to directions for future work.

2 Background and Motivation

2.1 Related Terminology

We start by introducing a few special terms that are useful when studying the literature on issues related to multidimensional databases.

OLAP: OLAP abbreviates *On-Line Analytical Processing*. As opposed to the well-known OLTP (On-Line *Transaction* Processing), focus is on data analyses rather than transactions. Furthermore, the analyses occur “On-Line”, i.e., fast, “interactive” query response is required. OLAP systems always employ a multidimensional view of data.

Data Warehouse: A data warehouse (DW) is a repository of integrated enterprise data that is used specifically for decision support, i.e., there is (typically, or ideally) only one data warehouse in an enterprise. The data in a DW is typically collected from a large number of sources within (and sometimes also outside) the enterprise.

Data Mart: A data mart (DM) is a subset of a data warehouse that is specialized for the needs of a special user group, e.g., the marketing department.

ETL: ETL (Extract-Transform-Load) is the three-step process that puts data into the DW. First, data is *extracted* from the operational source systems, e.g., ERP systems. Second, data is *transformed* from the source system formats into the DW format. This includes combining data from several sources and performing *cleansing* to correct errors such as missing or wrong data. Third, data is *loaded* into the DW. ETL is at times also referred to as ETT (Extract-Transform-Transport).

Business Intelligence: Business Intelligence (BI) is the process of making “intelligent” business decisions by analyzing available data. From a technological point of view, BI covers the combined areas of data warehousing, reporting, OLAP, data mining, some data visualization, what-if analysis, and special-purpose analytical applications.

2.2 Multidimensional History

Multidimensional databases do not have their origin in database technology, but stem from multidimensional matrix algebra, which has been used for (manual) data analyses since the late 19th century.

During the late 1960s, two companies, IRI and Comshare, independently began the development of systems that later turned into multidimensional database systems. The IRI Express tool became very popular in the marketing analysis area in the late 1970s and early 1980s; it later turned into a market-leading OLAP tool and was acquired by Oracle. Concurrently, the Comshare system developed into System W, which was used heavily for financial planning, analysis, and reporting during the 1980s.

In 1991, Arbor was formed with the specific purpose of creating “a multiuser, multidimensional database server”, which resulted in the Essbase system. Arbor, now Hyperion, later licensed a basic version of Essbase to IBM for integration into DB2. It was Arbor and Codd who, in 1993, coined the term OLAP [5].

Another significant development in the early 1990s was the advent of large *data warehouses* [22], which are typically based on relational *star* or *snowflake* schemas, an approach to implementing multidimensional databases using relational database technology.

In 1998, Microsoft shipped its MS OLAP Server, the first multidimensional system aimed at the mass market. This has led to the current situation where multidimensional systems are increasingly becoming commodity products that are shipped at no extra cost together with leading relational database systems. Since then, a number of open source BI tools have become available, for a survey see [48].

A more in-depth coverage of the history of multidimensional databases is available in the literature [49].

2.3 Spreadsheets and Relations

Let us assume that we want to analyze sales of products in a supermarket chain, for which we capture *what* product was sold, *when* it was sold, and *where* (in what store) it was sold, along with the number of items sold and the total sales price for those. When deciding what technologies to use to analyze such data, *spreadsheets* immediately come to mind as a possibility.

We can easily build a spreadsheet with the sales on individual days as rows, and the columns covering the products. However, capturing both number of items sold and sales price means we have to use two columns for every product. Furthermore, the formulae for computing cell values and adding them up will be duplicated for each cell. Immediately, the question that pops up is how to capture the sales location, the store where the product was sold. We could have a separate sheet for each store, but that is very cumbersome to handle, and it does not generalize to more than three dimensions, i.e., adding a fourth dimension like the customer buying the product would be infeasible. We thus miss the concept of true *multidimensionality*.

A further complication comes from *grouping the data*. We could imagine grouping products into product groups and days into weeks, but it will be very hard to generalize this to several levels of products, each level having attributes, and complex groupings of time, especially if we want several alternative ways of grouping for each dimension. If we introduce such concepts the mix of data and hierarchy definitions will often cause the latter to be duplicated. We thus miss the concept of *hierarchical dimensions*.

On top of this, it is almost impossible to add new types of data to the structure, e.g., adding the profit of the individual sale, structuring the product dimension into hierarchical levels, etc. Being database people, we see that we need to separate data and *schema*. Another aspect of this is that summing up data along rows or columns is easy, but with more complex aggregation formulae it becomes hard to manage, since the formulae have to be copied to a huge number of cells, making maintenance a nightmare. Also, computing subaggregates for months, quarters, and years is not easy to do. We thus need the notion of *automatic aggregation*.

With spreadsheets not meeting our requirements for managing multidimensional data, we could consider using an SQL database. Relational systems offer flexibility in the modeling and querying of data, separate schema and data, etc. However, the problem is that many important computations, including cumulative aggregates (sales in year to date), totals and subtotals together, and rankings (top 10 selling products), are hard or impossible to formulate in standard SQL. This is because *interrow* computations are difficult to express in SQL—only *intercolumn* computations are easy. Also, transpositions of rows and columns require cumbersome manual specifications and combinations of multiple views. Although extensions of SQL, such as the *data cube operator* [14] and *query windows* [13] can solve some of the problems, the concepts of true multidimensionality, hierarchical dimensions, and automatic aggregation are not well supported.

In summary, neither spreadsheets nor relational databases fully support our requirements for advanced data analyses, except in very restricted scenarios. As we will see shortly, multidimensional databases with OLAP applications offer all these features. Indeed, OLAP is often referred to as *spreadsheets on steroids*.

3 Multidimensional Concepts

We first offer an overview of the concept of a multidimensional cube, then cover dimensions, facts, and measures in turn.

3.1 Data Cubes

Data cubes provide true multidimensionality. They generalize spreadsheets to any number of dimensions. In addition, hierarchies in dimensions and formulas

are first-class, built-in concepts, meaning that these are supported without duplicating their definitions. A collection of related cubes is commonly referred to as a *multidimensional database* or a *multidimensional data warehouse*.

We obtain a higher dimensional cube for our sales example by including additional dimensions. The most pertinent example of an additional dimension is a time dimension, but it is also possible to include other dimensions, e.g., an artist dimension that describes the artists associated with albums. In a cube, the combinations of a dimension value from each dimension define the *cells* of the cube. The actual sales counts are stored in the corresponding cells.

In a cube, dimensions are first-class concepts with associated domains, meaning that the addition of new dimension values is easily handled. Although the term “cube” implies 3 dimensions, a cube can have any number of dimensions. It turns out that most real-world cubes have 4–12 dimensions [22, 49]. Although, there is no theoretical limit to the number of dimensions, current tools often experience performance problems when the number of dimensions is more than 10–15. To better suggest the high number of dimensions, the term “hypercube” is often used instead of “cube.”

Figure 1 illustrates a three-dimensional cube. Let us assumed that we have sales data for the years 2010 and 2011 for the products milk and bread in the cities of Aalborg and Copenhagen. We then have three dimensions: Time, Product, and Location.

| Location | Time | Product | |
|------------|------|---------|-------|
| | | Milk | Bread |
| Aalborg | 2011 | 1031 | 2543 |
| | 2010 | 2072 | 3984 |
| Copenhagen | 2011 | 16 | 43765 |
| | 2010 | | |

Fig. 1. Sales Data Cube

Depending on the specific application, a highly varying percentage of the cells in a cube are non-empty, meaning that cubes range from *sparse* to *dense*. Cubes tend to become increasingly sparse with increasing dimensionality and with increasingly finer granularities of the dimension values.

A non-empty cell is called a *fact*. The example has a fact for each combination of time, product, and city where at least one sale was made. A fact has associated with it a number of *measures*. These are numerical values that “live” within the cells. In the figure, we show just one measure, the total sales price, but we could also put the number of items sold into the cells.

Generally, only 2 dimensions (or sometimes 3 with simulated 3D views) can be viewed (and the correlations between them understood) at the same time, although for low-cardinality dimensions, a few more dimensions can be viewed by nesting one dimension within another on the axes. Thus, the dimensionality of a cube must be reduced at query time by *projecting* it down to fewer dimensions via *aggregation* of the measure values across the projected-out dimensions. For example, if we want to view just sales by City and Time, we aggregate over the entire dimension that characterizes the sales by Product for each combination of City and Time (in real-life, this will be many cells).

An important goal of multidimensional modeling is to “provide as much context as possible for the facts” [22]. The concept of *dimension* is the central means of providing this context. One consequence of this is a different view on *data redundancy* than in relational databases. In multidimensional databases, controlled redundancy is generally considered appropriate, as long as it considerably increases the information value of the data. One reason to allow redundancy is that multidimensional data is often *derived* from other data sources, e.g., data from a transactional relational system, rather than being “born” as multidimensional data, meaning that updates can more easily be handled [22]. However, there is usually no redundancy in the facts, only in the dimensions.

Having introduced the cube, we describe its principal elements, dimensions, facts, and measures, in more detail.

3.2 Dimensions

The notion of a dimension is an essential and distinguishing concept for multidimensional databases. Dimensions are used for two purposes: the *selection* of data and the *grouping* of data at a desired level of detail.

A dimension is organized into a containment-like hierarchy composed of a number of *levels*, each of which represents a level of detail that is of interest to the analyses to be performed. The instances of the dimension are typically called *dimension values*. Each such value belongs to a particular level.

In some cases, it is advantageous for a dimension to have *multiple hierarchies* defined on it. For example, a Time dimension may have hierarchies for both *Fiscal Year* and *Calendar Year* defined on it. Multiple hierarchies share one or more common lowest level(s), e.g., Day and Month, and then group these into multiple levels higher up, e.g., Fiscal Quarter and Calendar Quarter to allow for easy reference to several ways of grouping. Most multidimensional models allow multiple hierarchies. A dimension hierarchy is defined in the metadata of the cube, or the metadata of the multidimensional database, if dimensions can be shared. This means that the problem of duplicate hierarchy definitions as discussed in Section 2.3 is avoided.

In Figure 2, the schema and instances of a sample *Location* dimension for our cube data are shown. The Location dimension has three levels, the City level being the lowest. City level values are grouped into *Country* level values, i.e., countries. For example, Aalborg is in Denmark. The \top (“top”) level represents *all* of the dimension, i.e., every dimension value is part of the \top (“top”) value.

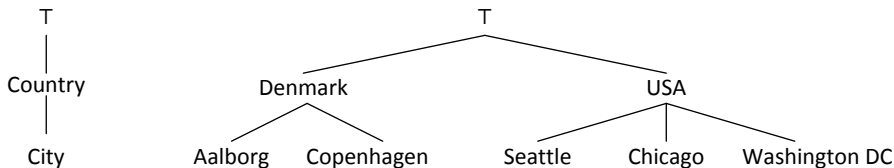


Fig. 2. Schema and Instance for the Location Dimension

In some multidimensional models, a level may have associated with it a number of *level properties* that are used to hold simple, non-hierarchical information. For example, the weight of a product (specifically, a Stock Keeping Unit (SKU)) can be a level property in the SKU level of the Product dimension. This information could also be captured using an extra Weight dimension. Using the level property has the effect of not increasing the dimensionality of the cube.

Unlike the linear spaces used in matrix algebra, there is typically no ordering and/or distance metric on the dimension values in multidimensional models. Rather, the only ordering is the containment of lower-level values in higher-level values. However, for some dimensions, e.g., the Time dimension, an ordering of the dimension values is available and is used for calculating cumulative information such as “total sales in year to date”.

Most models require dimension hierarchies to form *balanced trees*. This means that the dimension hierarchy must have uniform height everywhere, e.g., all departments, even small ones, must be subdivided into project groups. Additionally, direct links between dimension values can only go between immediate parent-child levels, and not jump two or more levels. For example, if all cities are first grouped into states and then into countries, cities cannot be grouped directly under countries (as is the case in Denmark, which has no states). Finally, each non-top value has precisely one parent, e.g., a product must belong to exactly one product group. In Section 4, we discuss the relaxation of these constraints.

3.3 Facts

Facts are the objects that represent the *subject* of the desired analyses, i.e., the interesting “thing”, or event or process, that is to be analyzed to better understand its behavior.

In most multidimensional data models, the facts are *implicitly* defined by their combination of dimension values. If a non-empty cell exists for a particular combination, a fact exists; otherwise, no fact exists. (Some other models treat facts as first-class objects with a separate identity [38].) Next, most multidimensional models require that each fact be mapped to precisely one dimension value at the lowest level in each dimension. Other models relax this requirement [38].

A fact has a certain *granularity*, determined by the levels from which its combination of dimension values are drawn. For example, the fact granularity in our example cube is “Year by Product by City.” Granularities consisting of

higher-level or lower-level dimension levels than a given granularity, e.g., “Year by Product Category by City” or “Day by Product by City” for our example, are said to be *coarser* or *finer* than the given granularity, respectively.

It is commonplace to distinguish among three kinds of facts: *event* facts, *state* facts, and *cumulative snapshot* facts [22]. Event facts (at least at the finest granularity) typically model *events in the real world*. The events describe an overall real-world process that is captured, e.g., sales for a supermarket chain. A unique instance of the process, e.g., a particular sale of a given product in a given store at a given time, is represented by one fact. Examples of event facts include sales, clicks on web pages (for web usage mining [21]), and the flow of goods in and out of (real) warehouses.

A snapshot fact models the *state* of a given process at a given point in time. Typical examples of snapshot facts include the inventory levels in stores and warehouses, and the number of users using a web site. For snapshot facts, the same object, e.g., a specific can of beans on a shelf, with which the captured real-world process, e.g., inventory management, is concerned, may occur in several facts at different time points.

Cumulative snapshot facts are used to handle information about *a process up to a certain point in time*. For example, we may consider the total sales in year to date as a fact. Then the total sales up to and including the current month this year can be easily compared to the figure for the corresponding month last year.

Often, all three types of facts can be found in a given data warehouse, as they support complementary classes of analyses. Indeed, the same base data, e.g., the movement of goods in a (real) warehouse, may often find its way into three cubes of different types, e.g., warehouse flow, warehouse inventory, and warehouse flow in year-to-date.

3.4 Measures

A *measure* has two components: a *numerical property* of a fact, e.g., the sales price or profit, and a *formula* (most often a simple aggregation function such as SUM) that can be used to combine several measure values into one. In a multidimensional database, measures generally represent the properties of the chosen facts that the users want to study, e.g., with the purpose of optimizing them.

Measures then take on different values for different combinations of dimension values. The property and formula are chosen such that the value of a measure is meaningful for all combinations of aggregation levels. The formula is defined in the metadata and thus not replicated as in the spreadsheet example. Although most multidimensional data models have measures, some do not. In these, dimension values are also used for computations, thus obviating the need for measures, but at the expense of some user-friendliness [38].

It is important to distinguish among three classes of measures, namely *additive*, *semi-additive*, and *non-additive* measures, as these behave quite differently in computations.

Additive measure values can be combined meaningfully along any dimension. For example, it makes sense to add the total sales over Product, Location, and Time, as this causes no overlap among the real-world phenomena that caused the individual values. Additive measures occur for any kind of fact.

Semi-additive measure values cannot be combined along one or more of the dimensions, most often the Time dimension. Semi-additive measures generally occur when the fact is of type snapshot or cumulative snapshot. For example, it does not make sense to sum inventory levels across time, as the same inventory item, e.g., a specific product, may be counted several times, but it is meaningful to sum inventory levels across products and stores.

Non-additive measure values cannot be combined along any dimension, usually because of the chosen formula. For example, this occurs for an “average sales price” measure. Here, the averages for lower-level values cannot be directly combined into averages for higher-level values. Non-additive measures can occur for any kind of fact.

3.5 Multidimensional Querying

We now briefly introduce multidimensional querying.

The most common query is *slice* which does a selection on a cube, thus reducing the cube. A *dice* views the cube from a different viewpoint (rotates it), e.g., by Year by City. The effects of these on a cube resemble preparing an onion for cooking, thus the names.

The query types known as *drill-down* and *roll-up* use the dimension hierarchies and measures to perform aggregations at given levels, and are the inverses of each other. Please consider the Location dimension in Figure 2 together with the cube, where the (additive) measure is the sales price together with the function SUM. When we roll-up from the City level to the Country level, the measure values for all cities in the same country are combined into one by the associated formula (summed). When we drill-down from the Country level to the City level, a cell for a given country is exploded into a number of cells, one for each city, i.e., dis-aggregating the data in the country cell.

These four operations may of course be combined and nested.

For a multidimensional database with several cubes that share one or more dimensions, the so-called *drill-across* operation combines the cubes via the shared dimensions, the analogue of a relational join.

Queries on order, which are very important for data analysis, are well supported by multidimensional databases. Such queries order cells in results, return only the top or bottom cells according to the specified order, and are often referred to as *ranking* or *TOP N/BOTTOM N* queries [49].

4 Complex Multidimensional Data

The traditional multidimensional data models and implementation techniques assume that the data being modeled conforms to a quite rigid regime. Specifically, it is typically assumed that all facts map (directly) to dimension values

at the lowest levels of the dimensions and only to one value in each dimension. Further, it is assumed that the dimension hierarchies are simply balanced trees. In many cases, this is adequate to support the desired applications satisfactorily. However, situations occur where these assumptions are too rigid for comfort.

In such situations, the support offered by “standard” multidimensional models and systems is inadequate, and more advanced concepts and techniques are called for. A more comprehensive treatment of complex multidimensional data is available in the literature [11]. We proceed to consider the impact of irregular hierarchies on pre-computation.

Complex multidimensional data is problematic as it is not summarizable [25, 46, 16]. Intuitively, data is *summarizable* if the results of higher-level aggregates can be derived from the results of lower-level aggregates. Without summarizability, users will either get wrong query results, if they base them on lower-level results, or computation may be prohibitively time consuming because we cannot use pre-computed lower-level results to compute higher-level results. When it is no longer possible to pre-compute, store, and subsequently reuse lower-level results for the computation of higher-level results, aggregates must instead be calculated directly from base data, which is what leads to the increased computational costs.

It has been shown that summarizability is preserved if the aggregate functions are distributive and the hierarchies of dimension values are *strict*, *onto*, and *covering* [25, 38]. Informally, a dimension hierarchy is *strict* if no dimension value has more than one (direct) parent, *onto* if the hierarchy is balanced, and *covering* if no containment path skips a level. Intuitively, this means that dimension hierarchies must be balanced trees. If this is not the case, some lower-level values will be either double-counted or not counted when reusing intermediate query results for the computation of other results.

Figures 3 and 4 contain two dimension hierarchies: a Location dimension hierarchy that includes a State level, and a Product dimension hierarchy that captures one possible categorization of products into categories. The hierarchy in Figure 3 is *non-covering* because Denmark has no states and because Washington DC belongs to no state. If we pre-compute aggregates at the State level, we will have no values for Aalborg, Copenhagen, and Washington DC, which has the effect that facts mapped to these cities will not be taken into account when computing country aggregates from pre-computed State level aggregates.

The hierarchy in Figure 4 is *non-onto* because the Vegetables product category has no further subdivision. If we materialize aggregates at the lowest level, facts mapping directly to Vegetables will not be counted. The hierarchy is also *non-strict* because the product Skimmed Milk is shared between categories Dairy and Diet. If we materialize aggregates at the middle level, data for Skimmed Milk will be used twice, for both Dairy and Diet, which is what we want at this level. However, this means that the data will also be used twice if we combine these aggregates into the grand total, i.e., if we reuse these aggregates for further aggregation.

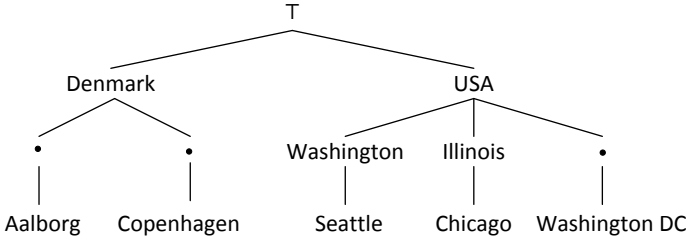


Fig. 3. Irregular Location Dimension

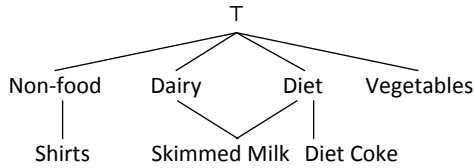


Fig. 4. Irregular Product Dimension

Irregular dimension hierarchies occur in many contexts, including organization hierarchies [56], medical diagnosis hierarchies [31], and concept hierarchies for web portals such as that of Yahoo! [55].

A solution to the problems with irregular hierarchies is to *normalize* the hierarchies, a process that pads non-onto and non-covering hierarchies with “dummy” dimension values to make them onto and covering, and fuses sets of parents in order to remedy the problems with non-strict hierarchies. This transformation may be accomplished transparently to the user, rendering pre-aggregation applicable to the more general types of dimension hierarchies discussed here [37]. An interesting proposal for studying the properties of irregular dimensions in a very generic and flexible way is the *dimension constraints* framework by Hurtado et al. [16].

5 Support for Complex Multidimensional Data

We proceed to present requirements for multidimensional data model support of complex multidimensional data. This is followed by a characterization of how existing models support these. The resulting survey is a condensed and updated version of an existing survey [38]. The survey provides an overview of the levels of complexity in the data that the different models support. Another survey of multidimensional models can be found in [54].

5.1 Requirements for Complex Multidimensional Data

We first describe requirements that a multidimensional data model should satisfy in order to support complex multidimensional data.

1. *Explicit hierarchies in dimensions.* The hierarchies in the dimensions should be captured explicitly by the schema. This permits the user to drill-down and roll-up, as discussed in Section 3.5.
2. *Multiple hierarchies in each dimension.* A single dimension can have several paths for data aggregation. As an example, assume that we have a Time dimension that captures both the Calendar Year and the Fiscal Year. To model this, multiple hierarchies are needed.
3. *Support for aggregation semantics.* The data model should capture the aggregation semantics of the data and use this to provide a “safety net” that catches queries that might yield results that have no meaning to the user. Aspects of this include built-in support for avoiding double-counting of data and avoiding addition of non-additive data.

For example, when asking for the number of products sold in different categories, Skimmed Milk sales will be counted as both Dairy sales and Diet Sales. However, if the user then adds up the category totals in an attempt to obtain the total number of sales, the system should prevent this, as Skimmed Milk sales will then be counted twice.

The user should also be able to specify which aggregations are considered meaningful for the different kinds of data available, and the model should provide a foundation for enforcing these specifications. As an illustration, it may not be meaningful to sum up inventory levels across time, while performing average calculations on them does make sense.

4. *Non-strict hierarchies.* As explained in Section 4 and illustrated in Figure 4, dimension hierarchies may be non-strict, i.e., we can have many-to-many relationships between the different levels in a dimension. Because such dimensions make sense to the users, they should be supported by the data model.
5. *Non-onto hierarchies.* As also covered in the previous section, dimension hierarchies may be unbalanced, i.e., the path from the root to a leaf may have varying length for different leaves. This is illustrated in Figure 4.
6. *Non-covering hierarchies.* Another common feature of real-world hierarchies is that links between two nodes in the hierarchy “skip” one or more levels, as in Figure 3.
7. *Symmetric treatment of dimensions and measures.* The data model should allow measures to be treated as dimensions and vice versa. In our case, the Sales attribute would typically be treated as a measure, to allow for computations such as total sales, etc., but we should also be able to define a SalesVolume dimension which allows us to group sales into groups such as small, medium, and large sales.
8. *Many-to-many relationships between facts and dimensions.* The relationship between fact and dimension does not always have the classical many-to-one cardinality. In a case for hospital patients, some patient may have more than one diagnosis [38]. In a supermarket case, a product could belong to several (partly overlapping) groups, e.g., Skimmed Milk belongs both to the Dairy and Diet product groups.

9. *Handling change and time.* Although data changes over time, it should be possible to perform meaningful analyses across times when data changes. For example, product hierarchies classifying products into categories change over time. So do Location hierarchies, where store districts change. It should be possible to easily combine data across changes. The problem of *slowly changing dimensions* [22] is part of this problem.
10. *Handling different levels of granularity.* Fact data might be registered at different granularities. For example, US sales might be reported per state, while Danish sales might be reported per city. It should still be possible to get correct analysis results when data is registered at different granularities.
11. *Handling imprecision.* Finally, it is very important to be able to capture the imprecision in the data directly and allow queries to take this into account. For example, the mapping of patient to diagnoses could have varying precision, as some diseases are easy to categorize exactly, while others are not [38].

Many other requirements may be posed to multidimensional data models. We have chosen the eleven requirements above for several reasons: First, they are non-trivial and are not satisfied by all existing models. Second, they are “model” requirements that affect the core of a multidimensional data model. This contrasts with less fundamental requirements that may be met by simply adding a new facility to the data model’s query language. Third, they derive from previous studies of the types of data and desired analyses that may be found in complex systems [34, 36].

5.2 Existing Multidimensional Models

We proceed to evaluate nineteen data models for data warehousing on the requirements just presented. We consider the models of Rafanelli & Shoshani [46], Agrawal et al. [2], Gray et al. [14], Dyreson [12], Kimball [22], Li & Wang [26], Gyssens & Lakshmanan [15], Cabbibo and Torlone [4], Datta & Thomas [6], Lehner [23], Vassiliadis [53], Jagadish et al. [17], Mendelzon & Vaisman [29], Pedersen et al. [35, 38], Abelló et al. [1], Boussaid et al. [3], Trujillo et al. [51], and Malinowski et al. [28], and Microsoft’s Analysis Services data model [50, 47, 30]. These models are good representatives of the prominent models in both the research community and commercial systems. The models can be divided into *simple cube models*, *structured cube models*, *complex cube models*, and *statistical object models*.

The simple cube models [6, 14, 15, 22] treat data as n -dimensional cubes. Generally, the data is divided into *facts* or *measures*, e.g., Sale, on which calculations should be performed, and *dimensions*, e.g., Product, which characterize the facts. Each dimension has a number of attributes, which can be used for selection and grouping. In our example, the Product dimension might have a Product Name attribute and a Category attribute that would be used to characterize the sales. The hierarchy between these attributes is not captured explicitly by the

schema of the simple cubes, so these models do not “know” that products roll up to categories.

Kimball’s star schema model is the best example of the simple cube models. His model is based on plain SQL and does not embody multidimensional concepts per se. We include it here because it is the most widely used implementation model for multidimensional databases. Additionally, most relational OLAP tools assume a star schema structure of the database and cannot handle more complex designs. Thus, the evaluation of the star schema model is based on what can be achieved using a plain star schema design and the corresponding (simple) SQL queries, not on what can be done using full-fledged SQL.

The structured cube models [2, 4, 12, 17, 23, 26, 29, 50, 47, 30, 53] capture the hierarchies in the dimensions explicitly, providing better guidance for the user navigating the cubes. This information may also be useful for query optimization [24]. The hierarchies are captured using either *grouping relations* [26], *dimension merging functions* [2], measure graphs [12], roll-up functions [4, 29], level lattices [53], hierarchy schemas and instances [17], or an explicit tree-structured hierarchy as part of the cube [23, 30, 47, 50].

The complex cube models contain the models by Pedersen et al. [35, 38], Abelló et. al. [1], Boussaid et. al. [3], Trujillo et al. [51], and Malinowski et. al. [28]. The Pedersen model was designed to support the eleven requirements presented above. The complex cube models generally contain constructs for handling at least a good deal of the requirements. Specifically, the dimensions have explicit schemas and levels (Req. 1 and 2), and the dimension instances allow even complex hierarchical structures (at least partial support for one or more of Req. 4, 5, and 6). With complex hierarchies, summarizability becomes an issue, so most of them contain an (at least partial) aggregation semantics framework to disallow erroneous computations (Req. 3). In the Pedersen model, dimensions and measures are treated equal, since everything is a dimension (Req. 7). For most of the models, the mappings between facts and dimensions allow facts to map to several values in the same dimension (Req. 8), and for some of the models also to map to dimension values in non-bottom levels (Req. 10). In the Pedersen model, mapping to non-bottom levels lays the foundation for handling imprecise data, which is further supported by the query algebra (Req. 11). Finally, in the Pedersen model (and partly in the Abelló model), dimension hierarchies and fact-dimensions relations are annotated with time validity information which is further supported by the query algebra, thus supporting Req. 9.

The last group of models is the *statistical object models* [46]. For this group, a structured classification hierarchy is coupled with an explicit aggregation function on a single measure to produce a “pre-cooked” object that will answer a very specific set of queries. This approach is not as flexible as the others, but unlike most of these, it provides some protection, by using aggregation semantics, against getting query results that are incorrect or not meaningful to the user.

The results of evaluating the nineteen data models against the eleven requirements are shown in Table II. If a model supports all aspects of a requirement, we say that the model provides *full* support, denoted by “√”. If a model supports

some, but not all, aspects of a requirement, we say that it provides *partial* support, denoted by “p”. When it has not been possible to determine how support for a requirement should be accomplished in the model, we say that the model provides *no* support, denoted by “-”.

Table 1. Evaluation of the Data Models

| Model | Requirement | | | | | | | | | | |
|-----------------------------------|-------------|---|---|---|---|---|---|---|---|----|----|
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| Rafanelli & Shoshani [46] | √ | - | √ | p | p | - | - | - | - | - | - |
| Agrawal et al. [2] | p | √ | - | p | - | - | √ | - | - | - | - |
| Dyreson [12] | √ | √ | p | - | - | - | - | - | - | p | p |
| Gray et al. [14] | - | √ | p | - | - | - | √ | - | - | - | - |
| Kimball [22] | - | √ | p | - | - | - | - | - | p | - | - |
| Li & Wang [26] | p | √ | p | - | - | - | - | - | - | - | - |
| Gyssens & Lakshmanan [15] | - | √ | p | - | - | - | √ | - | - | - | - |
| Cabbibo & Torlone [4] | √ | √ | p | - | - | - | - | - | - | - | - |
| Datta & Thomas [6] | - | √ | - | p | - | - | √ | - | - | - | - |
| Lehner [23] | √ | - | √ | - | - | - | - | - | - | - | - |
| Vassiliadis [53] | √ | √ | √ | - | - | - | - | - | - | - | - |
| Jagadish et al. [17] | √ | √ | - | - | √ | √ | - | - | - | √ | p |
| Mendelzon & Vaisman [29] | √ | √ | p | - | - | - | - | - | √ | - | - |
| Abelló et. al. [1] | √ | √ | √ | √ | - | - | √ | √ | p | √ | p |
| Boussaid et. al. [3] | √ | √ | - | p | p | - | √ | √ | - | p | p |
| Trujillo et. al. [51] | √ | √ | √ | √ | √ | √ | - | √ | p | - | - |
| Malinowski et. al. [28] | √ | √ | p | √ | √ | √ | - | - | p | - | - |
| Pedersen et al. [35, 38] | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ | √ |
| MS Analysis Services [30, 50, 47] | √ | √ | p | - | √ | √ | - | - | p | - | - |

1. *Explicit hierarchies in dimensions*: The simple cube models [6, 14, 15, 22] do not capture the hierarchies in the dimensions explicitly. Some models provide partial support via a *grouping relation* [26] and a *dimension merging function* [2], but do not capture the complete hierarchy together with the cube. This is done by the remaining models [4, 12, 17, 23, 29, 50, 30, 47, 35, 38, 1, 51, 28, 46, 53, 3], thus capturing the full cube navigation semantics in the schema.
2. *Multiple hierarchies in each dimension*: Some models [23, 46] require that the schema of dimension hierarchies is tree-structured. To support multiple hierarchies, a more general lattice structure is required. All the other models [2, 4, 6, 12, 14, 15, 17, 22, 26, 29, 50, 30, 47, 35, 38, 1, 51, 28, 53, 3] allow multiple hierarchies.
3. *Support for aggregation semantics*: Most of the models [4, 12, 14, 15, 22, 26, 29] support aggregation semantics partially, by implicitly requiring the dimension hierarchies to be *strict*, *onto*, and *covering*, i.e., the hierarchies should be balanced trees. This is one of the conditions of summarizability [25] and means that data will not be counted twice. Three of the models allow for

non-strict hierarchies, while not addressing the issue of double-counting, thus providing no support [2, 6, 3]. One model [17] allows non-onto and non-covering hierarchies, but does not address the issue of data not being counted, thus providing no support. One model [50, 30, 47] allows non-onto and non-covering hierarchies and provides mechanisms to count data correctly in these situations, thus providing partial support. Two models [23, 46] place explicit conditions on both the hierarchies (strict, onto, and covering) and the aggregation functions used (only additive data may be added, etc.), thus providing full support for aggregation semantics. One model [53] provides the support by always keeping a reference to the base data and computing from that when the aggregation semantics indicate the need to do so. One model allows non-strict hierarchies and provides aggregation semantics [1]. Finally, three models [35, 38, 51, 28] allows non-strict, non-onto, and non-covering hierarchies, with two models [35, 38, 51] providing full aggregation semantics, while one model [28] provides only partial aggregation semantics as it has no associated query algebra/language.

4. *Non-strict hierarchies*: Most of the models [4, 12, 14, 15, 17, 22, 23, 26, 29, 30, 50, 47, 53] implicitly or explicitly require that hierarchies be strict. Three models [2, 6, 3] either briefly mention that non-strict hierarchies are allowed, or seem to allow such hierarchies, but do not explore the issues raised by allowing this, e.g., the possibility of double-counting and the use of pre-computed aggregates. One model [46] investigates the possible problems with allowing non-strict hierarchies and advises against using this feature. Finally, four models [35, 38, 1, 51, 28] allows non-strict hierarchies, and prevents aggregation problems using (at least partial) aggregation semantics.
5. *Non-onto hierarchies*: One model [46] discusses the possibility of having non-onto hierarchies, but advises against using this feature. Six models [17, 30, 47, 50, 35, 38, 51, 28, 3] allow non-onto hierarchies either explicitly or implicitly. All the other models do not allow non-onto hierarchies.
6. *Non-covering hierarchies*: Only five models [17, 30, 47, 50, 35, 38, 51, 28] allow hierarchies to be non-covering. All the other models disallow non-covering hierarchies.
7. *Symmetric treatment of dimensions and measures*: Most of the models [4, 12, 17, 22, 23, 26, 29, 30, 50, 47, 46, 53, 28, 51] distinguish sharply between measures and dimensions. An attribute designated as a measure cannot be used as a dimensional attribute and vice versa. This restricts the flexibility of the cube designs, e.g., if the Sales attribute of the example is a measure, it cannot be used to group albums into sales groups. The other models [2, 6, 14, 15, 35, 38, 1, 3] do not impose this restriction. They either do not distinguish between measures and dimensions [14, 15, 3], allow for the conversion of measures to dimensions and vice versa [2, 6, 1], or treat all data as dimensions on which aggregate computations can also be performed [35, 38].
8. *Many-to-many relationships between facts and dimensions*: Only four of the models [35, 38, 1, 51, 3] allows many-to-many relationships between facts and their associated dimensions.

9. *Handling change and time*: Seven models [22, 29, 35, 38, 30, 47, 50, 1, 28, 51] fully or partially support this issue, but only the models of Mendelzon & Vaisman [29] and Pedersen et al. [35, 38] have built-in temporal support in the associated query algebra, thus fully supporting analyses across temporal changes in the dimensions. The other six models [22, 30, 47, 50, 1, 28, 51, 3] support slowly changing dimensions. None of the other models support analysis across changes.
10. *Handling different levels of granularity*: Dyreson [12] specifies an *incomplete data cube* to be a union of *cubettes*. Each cubette may have a different data granularity, thus providing some support for different levels of granularity. However, the granularity is fixed at the schema level, rather than at the data level, so the support is only partial. Four models [17, 35, 38, 1, 3] allow the granularity to vary at the data level. None of the other models handle different levels of granularity in the data.
11. *Handling imprecision*: For the reasons mentioned above, three models [12, 17, 1, 3] provide partial support for imprecision in the data, as varying granularities can provide a basis for handling imprecision. However, these models do not offer all the features necessary for handling the imprecision. One model [35, 38] provides full support for handling imprecision in the data. None of the other models provide explicit means for handling imprecise data.

To conclude, the models generally provide full or partial support for most of Requirements 1-3. Requirement 4 (non-strict hierarchies) is partially supported by three of the models and fully supported by four, while Requirement 5 (non-onto) is supported, partially or fully, by only six models. Requirement 6 is supported by five models, while Requirement 7 (symmetric treatment of dimensions and measures) is supported by six models. Requirement 8 (many-to-many fact-dimension relationships) is only supported by three models. Requirement 9 (handling change and time) is fully supported by two models and partially supported by another five. Requirement 10 (handling different levels of granularity) is partially supported by one model and fully supported by three, while Requirement 11 (imprecision) is partially supported by three models and fully by one.

6 Complex Application Domains

We now turn to describing complex multidimensional data as it appears in various application domains and what solutions can be applied to manage the complex data. Although presented for a particular domain, the solutions can be generalized to handle similar complexities in multidimensional data occurring in other domains.

6.1 Medical Data

The first domain is medical, or clinical, data, i.e., data about patients, diagnoses, symptoms, etc.

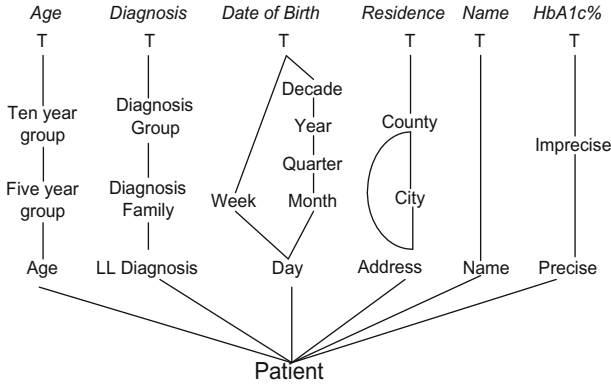


Fig. 5. Complex Multidimensional Schema (from [38])

Figure 5 shows the multidimensional schema of a patient case study. The users want to analyze the *number of patients* with certain characteristics (factors), such as diagnoses, age or date of birth, residence and long term blood sugar levels, in order to understand any correlations between these factors. Thus, the facts are individual Patients. Choosing other parts of the data as the facts, e.g., the individual diagnostications, would make this analysis awkward and difficult. The Date of birth dimension of the patient has the usual grouping of days, months, etc. We see that days can be grouped into both weeks and month, where only the latter can be grouped further into quarters, etc., i.e., there are *multiple* hierarchies in the same dimension. Sometimes, the users prefer not to think in terms of birthdates, but rather in terms of patient ages, thus a (derived) Age dimension is needed. Ages are grouped first into five year groups and then into ten year groups. The \top symbol denotes the top (ALL) level. The next dimension captures Diagnoses, the most precise being Lowlevel (LL) Diagnoses, which are then grouped into Diagnosis Families and finally into Diagnosis Groups. The Residence dimension captures the patient address, which can then be grouped into cities and counties. Some rural addresses are not in cities, and are thus directly part of a county, thus the hierarchy is *non-covering*. The Name dimension is flat with no hierarchies. Finally, the HbA1c% dimension captures an indicator for the long term blood sugar level of a patient. This dimension can be used both for grouping and for aggregate computation, thus showing the need for *symmetric treatment of dimensions and measures*. This value can be either a precise number or a more imprecise range of values, showing the need of handling *imprecision*.

Figure 6 shows an instance of this schema, projected only on the Diagnosis and Name dimensions. We see that one patient can have multiple diagnoses, thus inducing a *many-to-many relationship between facts and dimensions*. We also see that while some have low-level diagnoses, some also have (more imprecise) diagnoses at the diagnosis family and diagnosis group levels, i.e., the dimension have *different granularities*. Diagnosis family 14 is not further subdivided into

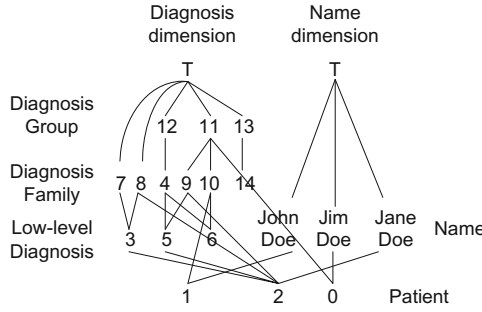


Fig. 6. Complex Multidimensional Instance (from [38])

low-level diagnoses (making the hierarchy *non-onto*) and low-level diagnosis 5 is a child of both 4 and 9 (making the hierarchy *non-strict*). The assignment of diagnoses changes over time, meaning that *change and time* must be handled to yield correct patient counts.

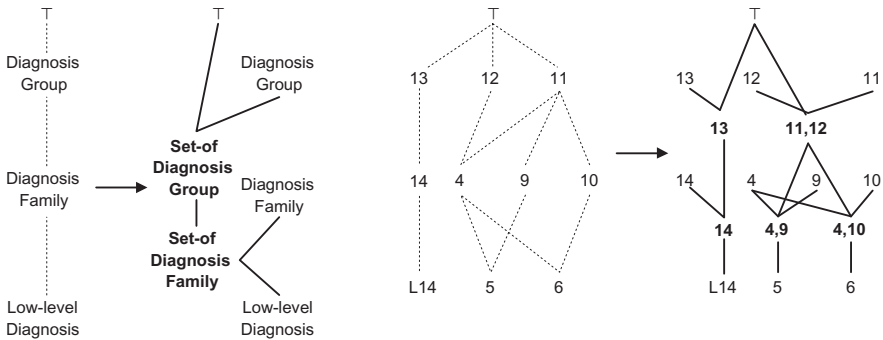


Fig. 7. Hierarchy Normalization (from [37])

The main problem with irregular hierarchies and many-to-many relationships between facts and dimensions is that summarizability is not preserved, which causes pre-aggregation not to work, as some parts will either be under-counted (in case of non-onto or non-covering hierarchies) while others will be over-counted (in case of non-strict hierarchies). The solution is to *normalize* the hierarchies to make them summarizable. The hierarchies are *padded* with the missing parts due to non-onto and non-covering hierarchies, e.g., hidden dummy low-level diagnoses are introduced. Non-strictness and many-to-many relationships are handled by *fusing* sets of parents into atomic set units and introduce a summarizable path to the top. This is illustrated in Figure 7 where the fusing process is shown. The transformation can be made transparent to the user by having separate *navigation* (shown to the user) and *aggregation* (used for computation) hierarchies. Further details can be found in [38, 37].

6.2 Spatio-temporal Data

Spatio-temporal data warehouses allowing the analysis of data related to both space and time have become increasingly important with the explosion of such data, e.g., from GPS devices. A characterization of the data and functionality in spatio-temporal data warehouses can be found in [52].

A key characteristic of spatial and spatio-temporal data is that overlaps, missing coverage, and partial containment often occur in hierarchies dividing a certain area according to one or more criteria. Figure 8 shows the schema and an instance of such a complex spatial hierarchy. Roadways are *partially* contained in Districts, as indicated by the dotted line in the schema. For example, Roadway is 40% contained in District3 (indicated by the weight 0.4). Districts are again partially contained in cities. For example, District2 belongs 30% to City1 and 70% to City2. Facts A, B, C, E, and G map to any level in the hierarchy, e.g., the position of C is only known at the city level, while for A it is known at the roadway level.

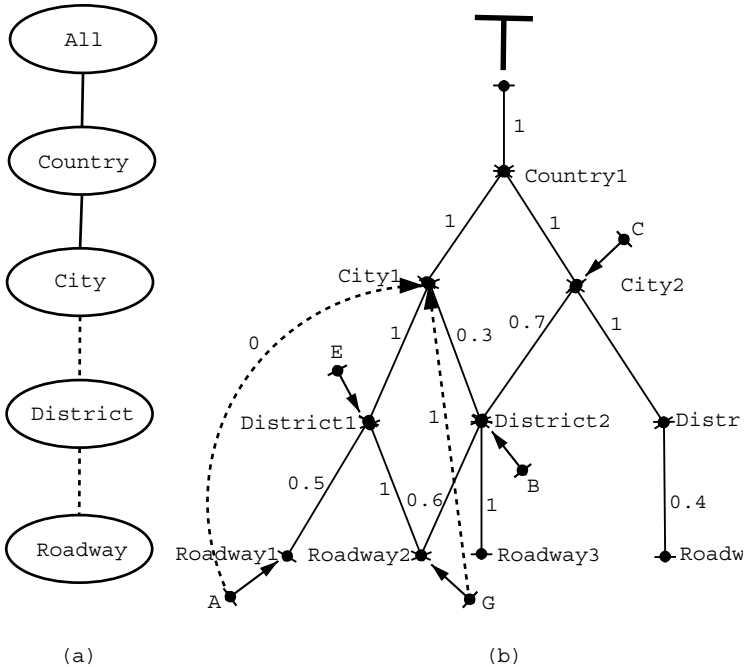


Fig. 8. Complex Spatial Hierarchy (from [18])

In order to make such a hierarchy summarizable and be able to apply pre-aggregation, a *normalization* procedure extended to handle partial containment, can be performed, see [18] for details.

6.3 Music Data

We now turn to the rather different application area of music data. Here, the aim is to provide efficient retrieval of songs based on both a set of given metadata criteria, e.g., artist, genre, year, etc., and retrieve songs that are *similar* to a given so-called seed song. For the metadata part, creating a number of metadata dimensions is a good fit. Figure 9 shows a complex Genre Dimension. We first divide songs into two overall genres, Pop and Rock. However, a fusion subgenre like PopRock then becomes a child of both of these, thus introducing *non-strictness* in the dimension. A way of dealing with this is to introduce *bitmap indices* for capturing which songs belong to which genres. These can efficiently be combined (Boolean AND/OR/NOT) to answer queries on multiple criteria. We see that songs 1 and 3 are Pop, while 2 and 3 are Rock, and 3 is PopRock. By storing the bitmaps for all dimension values, the non-strictness can be handled without problems. When the number of dimension values becomes large, i.e., many genres, the bitmap indices must be compressed using techniques like Position List Word Aligned Hybrid (PLWAH) [10].

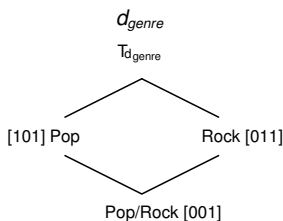


Fig. 9. Complex Genre Dimension and Bitmap (adapted from [19])

Another complexity occurs when capturing the similarities between songs. We need to capture the similarities between each of the n songs and all the other songs, i.e., n^2 distances. In many other applications, this can be done efficiently using various tree structures, because the distance function is metric, e.g., Euclidian distance. However, typical music similarity functions are *non-metric* as the triangular inequality does not hold [20]. Thus, we need to somehow store all the n^2 distances. However, it is not needed to store the exact distance values, instead only a partition into the most similar songs, the little less similar songs, and so on until the least similar songs suffices. This leads to the development of the so-called *distance store bitmap index* which uses compressed bitmaps to store these [20]. Figure 10 shows such a distance store with three partitions (distance from 1–3, from 3–6, and greater than 6) and the bitmaps capturing which songs belong to each. It is possible to use these to very efficiently search similar songs, filter out songs recently listened to, and combine with metadata criteria, all using very fast bitmap operations.

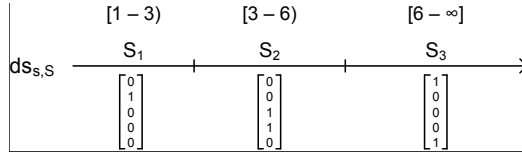


Fig. 10. Distance Store Bitmap Index (from [20])

6.4 XML Data

We now look at the problem of integrating an existing cube with *external XML data*. For example, let us assume that we have a cube about our sales of electronic components that we, due to a sudden need, want to extend with descriptions of these components available in an XML document on the web. We do not have time to go through the normal ETL procedure, but instead we want to bring in the external data *on demand*. We can do this using a so-called OLAP-XML Federation, as shown in Figure [11]. The OLAP data is accessed using the OLAP component, while access to the XML data is handled by the XML Component which wraps the XML sources to allow for easy querying. The Federation Manager uses its Metadata (sources, schemas, etc.) and Link Data (what dimension values map to which XML values), along with data retrieved from the OLAP and XML components to compute the query results, possible using Temp Data for intermediate storage. This allows for effective integration of external XML data for use as both dimensions and measures, see [41, 39] for details.

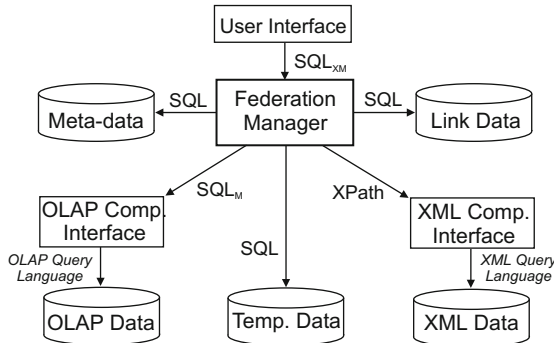


Fig. 11. OLAP-XML Federation (from [41])

One of the most important operations of the OLAP-XML Federation is the so-called *decoration* operator which “decorates” a cube with a new dimension. Figure [12] shows the new dimension resulting from decorating our sales cube with a new EC Description dimension. Since we use the so-called ALL semantics [40], several parents are possible, introducing *non-strictness*. For example, EC1234 is

both a D-type flip-flop and a 16-bit flip-flop. The operator is described in detail in [40]. A similar operator is *extension* which adds a new measure to the cube, based on external data [39].

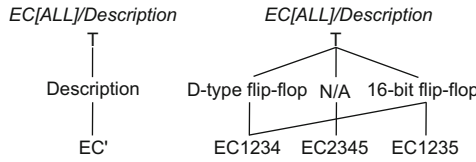


Fig. 12. Decoration With ALL Semantics (from [40])

6.5 Semantic Web Data

In recent years, another type of web data has become very popular, namely *semantic web data* which aims to add more semantics to data on the web using so-called ontologies. The most popular ontology language is called OWL which has several levels, the simplest being OWL Lite. Semantic web data has the form of *(Subject, Predicate, Object) triples* and are often stored in dedicated storage engines called *triplestores*. However, it is often desired to be able to combine triple data with other types of data, typically stored in an RDBMS, for advanced analytics purposes. Previous DBMS-based triplestores have suffered from insufficient scalability, while file-based triplestores are scalable but unable to integrate with RDBMS data. The 3XL triplestore [27] provides the best of both worlds by utilizing object-relational features, advanced buffering, and bulk-loading techniques to provide performance comparable to leading

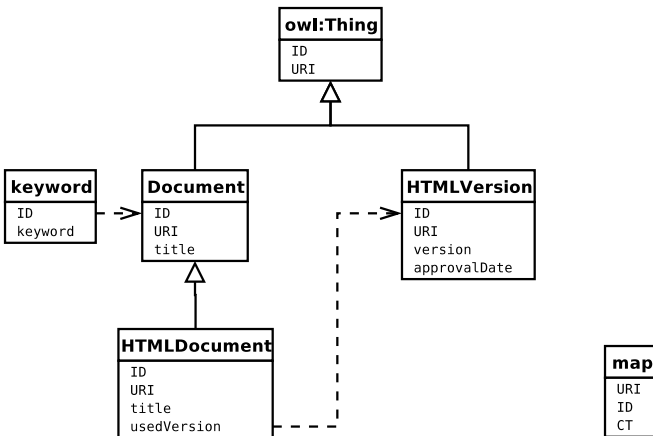


Fig. 13. 3XL Specialized Schema (from [27])

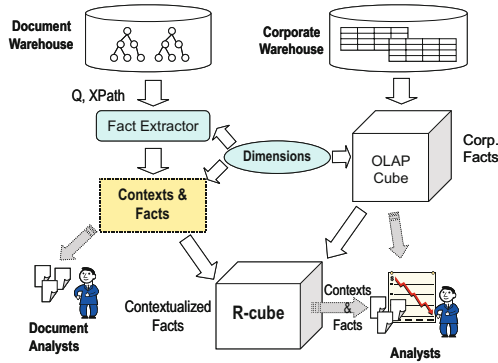


Fig. 14. Contextualized Data Warehouse (from [43, 45])

file-based triplestores while still allowing easy integration with other RDBMS data. 3XL uses the OWL Lite ontology for the triple data to create a specialized schema, see Figure 13. Here, object-relational table inheritance is used to capture that everything is an owl:Thing which is then specialized into Documents, which are again specialized into HTMLDocuments. The arrows indicate inheritance and the dotted lines indicate (soft) foreign keys. A special *map* table is maintained externally in BerkeleyDB to efficiently map URIs to short internal IDs and their associated class table (CT). Further details are found in [27].

6.6 Text Data

Finally, we will look at integration of multidimensional data with perhaps the most common type of data, namely *text*. Here, the problem is how to combine structured cube data like dimensions and measures with the unstructured world of text. One solution is the so-called *contextualized data warehouse* [42, 43, 45]. The architecture is shown in Figure 14. Data is assumed to be stored in two places, a structured corporate DW, and a semi-structured document warehouse (allowing the documents to have some structure, like article, section, paragraph, etc.). The multidimensional data is kept in a multidimensional cube. Using a user keyword query and a possible document structure query, relevant text sentences are extracted from the document warehouse. The Fact Extractor then tries to extract facts from the text, e.g., about locations and markets, products, times, etc. This is done by trying to find matches for the dimension values in the cube. The more dimensions a fact matches, the better relevance, e.g., if the fact matches both the Time, Product, and Location dimensions, the match is better than if it just matches one of them. Facts that arise from a single sentence are preferred to facts found in longer pieces of text. Matches can also be hierarchical, e.g., a city name matching through a country to a region. The closer the match is in the hierarchy (no. levels apart), the better. The so-called *relevances* of the extracted fact for cube facts are then computed, resulting in a ranking of which

documents are most closely linked to which facts. Thus, the documents have now been used to provide a context for the facts, i.e., *contextualizing* the cube with documents. The resulting cube structure is called a *relevance cube (R-cube)*.

7 Conclusion and Future Work

Multidimensional database concepts such as cubes, dimensions with hierarchies, and measures have become a cornerstone of business intelligence. However, as this lecture has shown, the standard data models and system implementations (OLAP) for multidimensional databases are sometimes not able to capture the complexities of advanced real-world application domains. This lecture showed how to manage such complex multidimensional data, including complex dimension hierarchies, complex measures, and integration of multidimensional data with complex external data. The lecture looked at how complex multidimensional data emerge in complex application domains such as medical data, location-based services, music data, web data, and text data, and presented solutions for these domains that support multidimensional business intelligence.

In future work, a number of important challenges must be addressed to support complex multidimensional data from emerging data sources and applications. These include warehousing data about the physical world, integrating structured, semi-structured, and unstructured data in DWs, integrating the past, the present, and the future, warehousing imperfect data, and ensuring privacy in DWs. Further challenges arise from the emerging domain of *cloud intelligence* [33] where support for massive scalability, data outsourcing, and collaborative BI becomes essential. The common base for addressing these challenges should be a new kind of data model, inspired by multidimensional and semi-structured data models, but capable of supporting a much wider range of data. Specifically, support will be added for handling geo-related data (geo models, etc), sensor data (high speed data streams, missing or incorrect values, etc), semi-structured and unstructured data (enabling analysis across structured, semi-structured, and unstructured data), and imperfect (imprecise, uncertain, etc.) data. Support for privacy management will also be built into the framework. Details can be found in [32, 33].

References

1. Abelló, A., Samos, J., Saltor, F.: YAM2: a multidimensional conceptual model extending UML. *Information Systems* 31(6), 541–567 (2006)
2. Agrawal, R., Gupta, A., Sarawagi, S.: Modeling multidimensional databases. In: ICDE, pp. 232–243 (1997)
3. Boussaid, O., Boukraa, D.: Multidimensional Modeling of Complex Data. In: Wang, J. (ed.) *Encyclopedia of Data Warehousing and Mining*, 2nd edn. (2008)
4. Cabibbo, L., Torlone, R.: Querying Multidimensional Databases. In: Cluet, S., Hull, R. (eds.) *DBPL 1997*. LNCS, vol. 1369, pp. 319–335. Springer, Heidelberg (1998)

5. Codd, E.F.: Providing OLAP (on-line analytical processing) to user-analysts: An IT mandate. E.F. Codd and Assoc. (1993)
6. Datta, A., Thomas, H.: A conceptual model and algebra for on-line analytical processing in decision support databases. In: WOITS, pp. 91–100 (1997)
7. Deliège, F., Chua, B.Y., Pedersen, T.B.: High-Level Audio Features: Distributed Extraction and Similarity Search. In: ISMIR, pp. 565–570 (2008)
8. Deliège, F., Pedersen, T.B.: Fuzzy Song Sets for Music Warehouses. In: ISMIR, pp. 21–26 (2007)
9. Deliège, F., Pedersen, T.B.: Using Fuzzy Lists for Playlist Management. In: Satoh, S., Nack, F., Etoh, M. (eds.) MMM 2008. LNCS, vol. 4903, pp. 198–209. Springer, Heidelberg (2008)
10. Deliège, F., Pedersen, T.B.: Position list word aligned hybrid: optimizing space and performance for compressed bitmaps. In: EDBT, pp. 228–239 (2010)
11. Dyreson, C.E., Pedersen, T.B., Jensen, C.S.: Incomplete information in multidimensional databases. In: Rafanelli, M. (ed.) Multidimensional Databases: Problems and Solutions. Idea Group Publishing (2003)
12. Dyreson, C.E.: Information retrieval from an incomplete data cube. In: VLDB, pp. 532–543 (1996)
13. Eisenberg, A., Melton, J.: SQL standardization: The next steps. SIGMOD Record 29(1), 63–67 (2000)
14. Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Venkatrao, M., Reichart, D., Pellow, F., Pirahesh, H.: Data cube: A relational aggregation operator generalizing group-by, cross-tab and sub-totals. DMKD 1(1), 29–54 (1997)
15. Gyssens, M., Lakshmanan, L.V.S.: A foundation for multi-dimensional databases. In: VLDB, pp. 106–115 (1997)
16. Hurtado, C.A., Gutiérrez, C., Mendelzon, A.O.: Capturing summarizability with integrity constraints in OLAP. TODS 30(3), 854–886 (2005)
17. Jagadish, H.V., Lakshmanan, L.V.S., Srivastava, D.: What can hierarchies do for data warehouses? In: VLDB, pp. 503–541 (1999)
18. Jensen, C.S., Kligys, A., Pedersen, T.B., Timko, I.: Multidimensional data modeling for location-based services. VLDBJ 13(1), 1–21 (2004)
19. Jensen, C.A., Mungure, E.M., Pedersen, T.B., Sørensen, K.: A Data and Query Model for Dynamic Playlist Generation. In: ICDE Workshops, pp. 65–74 (2007)
20. Jensen, C.A., Mungure, E.M., Pedersen, T.B., Sørensen, K., Deliège, F.: Effective Bitmap Indexing for Non-metric Similarities. In: Bringas, P.G., Hameurlain, A., Quirchmayr, G. (eds.) DEXA 2010, Part I. LNCS, vol. 6261, pp. 137–151. Springer, Heidelberg (2010)
21. Jespersen, S.E., Thorhauge, J., Pedersen, T.B.: A Hybrid Approach to Web Usage Mining. In: Kambayashi, Y., Winiwarter, W., Arikawa, M. (eds.) DaWaK 2002. LNCS, vol. 2454, pp. 73–82. Springer, Heidelberg (2002)
22. Kimball, R.: The Data Warehouse Toolkit. Wiley Computer Publishing (1996)
23. Lehner, W.: Modeling Large Scale OLAP Scenarios. In: Schek, H.-J., Saltor, F., Ramos, I., Alonso, G. (eds.) EDBT 1998. LNCS, vol. 1377, pp. 153–167. Springer, Heidelberg (1998)
24. Lehner, W., Ruf, T.: A redundancy-based optimization approach for aggregation in multidimensional scientific and statistical databases. In: DASFAA, pp. 253–262 (1997)
25. Lenz, H., Shoshani, A.: Summarizability in OLAP and statistical data bases. In: SSDBM, pp. 39–48 (1997)
26. Li, C., Wang, X.S.: A data model for supporting on-line analytical processing. In: CIKM, pp. 81–88 (1996)

27. Liu, X., Thomsen, C., Pedersen, T.B.: 3XL: Supporting efficient operations on very large OWL Lite triple-stores. *Information Systems* 36(4), 765–781 (2011)
28. Malinowski, E., Zimányi, E.: Hierarchies in a multidimensional model: From conceptual modeling to logical representation. *Data and Knowledge Engineering* 59(2), 348–377 (2006)
29. Mendelzon, A.O., Vaismann, A.A.: Temporal queries in OLAP. In: *VLDB*, pp. 242–253 (2000)
30. Microsoft. Microsoft SQL server: Analysis services, <http://www.microsoft.com/sql/technologies/analysis/default.mspx> (Current as of March 26, 2012)
31. National Health Service. Read Codes version 3. NHS (1999)
32. Pedersen, T.B.: Warehousing The World: A Vision for Data Warehouse Research. *Annals of Information Systems, Special Issue: New Trends in Data Warehousing and Data Analysis*, 1–17 (2009)
33. Pedersen, T.B.: Research challenges for cloud intelligence: invited talk. In: *EDBT/ICDT Workshops* (2010)
34. Pedersen, T.B., Jensen, C.S.: Clinical data warehousing—a survey. In: *MEDICON*, p. 20.3 (1998)
35. Pedersen, T.B., Jensen, C.S.: Multidimensional data modeling for complex data. In: *ICDE*, pp. 336–345 (1999)
36. Pedersen, T.B., Jensen, C.S.: Research issues in clinical data warehousing. In: *SSDBM*, pp. 43–52 (1999)
37. Pedersen, T.B., Jensen, C.S., Dyreson, C.E.: Extending practical pre-aggregation in on-line analytical processing. In: *VLDB*, pp. 663–674 (1999)
38. Pedersen, T.B., Jensen, C.S., Dyreson, C.E.: A foundation for capturing and querying complex multidimensional data. *Information Systems* 26(5), 383–423 (2001)
39. Pedersen, D., Pedersen, J., Pedersen, T.B.: Integrating XML Data in the TARGIT OLAP System. In: *ICDE*, pp. 778–781 (2004)
40. Pedersen, D., Pedersen, T.B., Riis, K.: The Decoration Operator: A Foundation for On-Line Dimensional Data Integration. In: *IDEAS*, pp. 357–366 (2004)
41. Pedersen, D., Riis, K., Pedersen, T.B.: XML-Extended OLAP Querying. In: *SSDBM*, pp. 195–206 (2002)
42. Pérez, J.M., Berlanga Llavori, R., Aramburu Cabo, M.J., Pedersen, T.B.: A Relevance-Extended Multi-dimensional Model for a Data Warehouse Contextualized with Documents. In: *DOLAP*, pp. 19–28 (2005)
43. Pérez, J.M., Berlanga Llavori, R., Aramburu Cabo, M.J., Pedersen, T.B.: R-Cubes: OLAP Cubes Contextualized with Documents. In: *ICDE*, pp. 1477–1478 (2007)
44. Pérez, J.M., Berlanga Llavori, R., Aramburu Cabo, M.J., Pedersen, T.B.: Integrating Data Warehouses with Web Data: A Survey. *IEEE TKDE* 20(7), 940–955 (2008)
45. Pérez-Martínez, J.M., Berlanga Llavori, R., Aramburu Cabo, M.J., Pedersen, T.B.: Contextualizing data warehouses with documents. *Decision Support Systems* 45(1), 77–94 (2008)
46. Rafanelli, M., Shoshani, A.: Storm: A Statistical Object Representation Model. In: Michalewicz, Z. (ed.) *SSDBM 1990*. LNCS, vol. 420, pp. 14–29. Springer, Heidelberg (1990)
47. Spofford, G., Harinath, S., Webb, C., Huang, D.H., Civardi, F.: *MDX-Solutions: With Microsoft SQL Server Analysis Services 2005 and Hyperion Essbase*. Wiley (2006)
48. Thomsen, C., Pedersen, T.B.: A Survey of Open Source Tools for Business Intelligence. *International Journal of Data Warehousing and Mining* 5(3), 56–75 (2009)

49. Thomsen, E.: OLAP Solutions: Building Multidimensional Information Systems. Wiley (1997)
50. Thomsen, E., Spofford, G., Chase, D.: Microsoft OLAP Solutions. Wiley (1999)
51. Trujillo, J., Palomar, M., Gómez, J., Song, I.-Y.: Designing Data Warehouses with OO Conceptual Models. *IEEE Computer* 34(12), 66–75 (2001)
52. Vaisman, A., Zimányi, E.: What Is Spatio-Temporal Data Warehousing? In: Pedersen, T.B., Mohania, M.K., Tjoa, A.M. (eds.) *DaWaK 2009*. LNCS, vol. 5691, pp. 9–23. Springer, Heidelberg (2009)
53. Vassiliadis, P.: Modeling multidimensional databases, cubes, and cube operations. In: *SSDBM*, pp. 53–62 (1998)
54. Vassiliadis, P., Sellis, T.K.: A survey of logical models for OLAP databases. *SIGMOD Record* 28(4), 64–69 (1999)
55. Yahoo! Yahoo!, <http://www.yahoo.com> (Current as of March 27, 2012)
56. Zurek, T., Sinnwell, M.: Data warehousing has more colours than just black and white. In: *VLDB*, pp. 726–729 (1999)

An Introduction to Business Process Modeling

Alejandro Vaisman

Université Libre de Bruxelles
avaisman@ulb.ac.be

Abstract. Business Process Modeling (BPM) is the activity of representing the processes of an organization, so that they can be analyzed and improved. Nowadays, with increased globalization, BPM techniques are used, for example, to optimize the way in which organizations react to business events, in order to enhance competitiveness. Starting from the underlying notion of workflow modeling, this paper introduces the basic concepts of modeling and implementing business processes using current information technologies and standards, such as Business Process Modeling Notation (BPMN) and Business Process Execution Language (BPEL). We also address the novel, yet growing, topic of Business Process Mining, and point out to open research challenges in the area.

1 Introduction

Business process management (BPM) comprises a collection of methods, techniques, and tools to support the design, management, analysis and execution of operational business processes [17]. BPM builds on classical workflow management (WFM) systems, although we can find its roots in office information systems [6,9] which used variants of Petri nets [11] to model office procedures. Although little advances were made in the eighties, the interest for workflow technology increased again in the nineties. Further, a standardization process led to the current BPMN 2.0 standard [1], released by the Object Management Group (OMG) [2]. In this paper we focus on business process modeling, providing a state-of-the-art in the topic, and showing the relationship between the most common modeling tools and formal methods like Petri nets, including an overview of scientific and practical issues.

Business process management systems (BPMS) follow current trends in software development, namely: (a) assembling complex software systems rather than coding from scratch, i.e., *orchestrating* pieces of software rather than coding individual modules; (b) moving the focus from data to processes. On the other hand, traditional software design was data-driven, i.e., data modeling was the starting point for building an information system. BPMS can be used to support this shift from programming to assembling, also supporting the notions of process orientation and redesign. For example, today's workflow management systems can be

¹ <http://www.omg.org/spec/BPMN/2.0/>

² <http://www.omg.org>

used to integrate existing applications and support process change just by means of changing the workflow diagram. This is also consistent with the development of applications based on web services: web services composition languages such as BPEL4WS³, WSCI⁴, and WSFL⁵ can be used to glue services defined using web services definition language (WSDL). Summarizing, BPMS are generic software systems driven by process design, used to manage operational business processes. In this sense, we can say that they are similar to database management systems (DBMS). Leading enterprise resource planning systems (ERP) (i.e., tailor-made systems) also offer a workflow management module. The workflow engines of SAP⁶ or PeopleSoft⁷ can be considered as integrated BPMS.

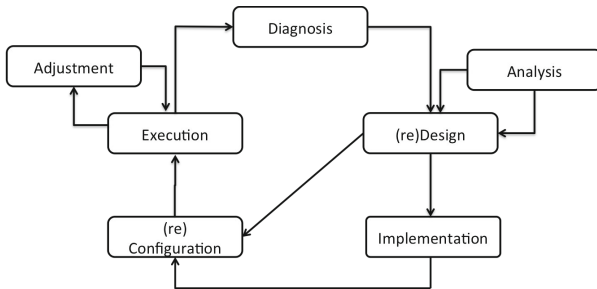


Fig. 1. BPM lifecycle

Figure 1 (adapted from [14]) depicts the BPM's life-cycle. We choose this as representative of many proposals found in the literature. In the figure we can identify the following phases: (a) *design*, where the processes are designed or re-designed; (b) *configuration*, where designs are turned into code; (c) *execution*, where business processes are executed using the configured system; (d) *diagnosis*, where the operational processes are analyzed to identify problems and to find things that can be improved. Note that phases (a) and (c) have associated cyclic tasks, namely *analysis* and *adjustment*, respectively.

This paper provides a general introduction to BPM, with focus in modeling, also providing the workflow notions that underlie business process (BP) modeling. We start introducing basic notions on workflow management and BP management and modeling (Section 2), discussing the similarities and differences between them, also introducing the terminology that will be used in the remainder. After this, we study formal methods for defining workflows (basically, Petri nets) (Section 3). In Section 4 we introduce the current OMG standard BPMN 2.0 (Business Process Modeling and Notation), and discuss its main constructs. In Section 5 we study tools for executing business process models, namely BPEL

³ <http://www.ibm.com/developerworks/webservices/library/ws-bpelcol1/>

⁴ <http://www.w3.org/TR/wsci/>

⁵ <http://www.ebpm1.org/wsfl.htm>

⁶ <http://www.sap.com/ERP>

⁷ <http://www.oracle.com/us/products/applications/peoplesoft-enterprise>

(Business Process Execution Language), and the translation from BPMN models to BPEL specifications. Section 6 briefly introduces other methods for specifying BP (with focus on UML Activity diagrams), and compares them against each other and against BPMN. Section 7 introduces the novel yet growing area of process mining, and in Section 8 we present some research topics in the field. We conclude in Section 9.

2 Workflow Management and Business Process Management and Modeling

We start this study providing some definitions that will give a precise framework for the remainder. We then discuss similarities and differences between workflows and business processes. The next definitions correspond to the ones given by the Workflow Management Coalition [3].

Definition 1 (Workflow). *We denote by Workflow the automation of a business process, in whole or part, during which documents, information or tasks are passed from one participant to another for action, according to a set of procedural rules.* □

Definition 2 (Workflow Management System). *A Workflow Management System (WFMS) is a system that defines, creates and manages the execution of workflows through the use of software, running on one or more workflow engines, and is able to interpret the process definition, interact with workflow participants and, when required, invoke the use of Information Technology tools and applications.* □

Definition 3 (Workflow System). *A Workflow System (WFS) is a system based on a WFMS that supports an specific set of business processes through the execution of computerized process definitions.* □

Note that the definitions above emphasize the focus on the use of software to support the execution of operational processes, also called enactment in workflow terminology. At the beginning of the twenty-first century, many researchers and practitioners realized that the traditional focus on execution was too restrictive, and terms like BPM were coined. There exist many definitions of BPM which, in general, refer to workflow management.

Definition 4 (Process Model). *A process model is a formalized view of a business process, represented as a coordinate set of parallel and/or sequential set of process activities that are connected to achieve a common goal.* □

Definition 5 (Business Process Modeling (cf. [17])). *Business Process Modeling is a modeling technique for supporting business processes using methods, techniques, and software to design, enact, control, and analyze operational processes involving humans, organizations, applications, documents and other sources of information.* □

⁸ WfMC, <http://www.wfmc.org/>

Definition 6 (Business Process Management System). *A generic software system designed to enact and manage operational business processes. This system should be process-aware and generic in the sense that it could be possible to modify the processes it supports. The process designs are often graphical and the focus is on structured processes that need to handle many cases.* □

BPM can be considered a natural evolution of workflow management, that enhances the latter with new BP technology, covering three process categories: interactions between (i) people-to-people; (ii) systems-to-systems; and (iii) systems-to-people, all from a process-centric perspective⁹. Workflow and business re-engineering are at the basis of BPM.

Although the term BPM usually refers to large projects in also large corporations, and ‘workflow’ products target more limited projects, the overlapping between both terms makes it impossible to draw a hard line between them. However, it is worth mentioning that, from a commercial point of view, workflow vendors are calling their systems as BPMS, in particular since the OMG adopted BPMN as the modeling standard language.

From a lifecycle point of view there are also overlaps and differences. Taking into account the lifecycle of Figure 11, the focus of traditional WFMS is on the first stages of such lifecycle, with little support for the *diagnosis* phase. Even the support for the *design* phase is limited to provide an editor and analysis tools, while real design support is missing. For example, few WFMS support simulation, verification, and validation of process design, although formalisms like Petri nets would allow this.

3 Using Workflows to Model Business Processes

Using graphical representations to allow a BP to be understood by the various stakeholders involved, is a normal practice in software design and BPM. In addition, it provides a unified vocabulary to reduce the risk of misunderstanding the problem at hand. In BPM, process models can be quite complex. Thus, using a formal language for their specification is a possible way of avoiding the problems above. Moreover, at the implementation level, the behavior of a BP can be explained in terms of the formal semantics of the specification language. The lack of a formal semantics (only partially solved with the advent of the BPMN 2.0 specification) has resulted in different interpretations by vendors for even basic control flow constructs. Further, definitions in natural language such as the ones provided by the Workflow Management Coalition are not precise enough.

To reduce the risk of costly corrections due to errors found at the final stages of a project, a thorough analysis of a BP must be carried out. For this kind of analysis, formal languages may play a key role. One formalism used to specify workflows are Petri nets [11]. The reasons for this are [19]: (a) Petri nets are formal; (b) they have associated analysis techniques; (c) they are state-based

⁹ Jon Pyke, former CTO of Staffware, contributor of BPTrends;

<http://www.bptrends.com>

rather than event-based. The fact that Petri nets cannot specify certain control flow dependencies led to the development of YAWL (standing for Yet Another Workflow Language) [16], whose formal semantics is specified as a transition system. We briefly discuss YAWL later in this paper. Although highly expressive, both formalisms are not very suitable for high-level specification, since they are not easy to understand to a typical user. Therefore, other techniques have been adopted, leading to the BPMN 2.0 standard, which we also address later in the paper. In the next section we introduce Petri nets, since they underlie BP modeling theory and practice.

3.1 Workflows in Action

Basic Terminology. The basic element a workflow system deals with is a so-called *case*. For example, an insurance claim (i.e., an instance of a process that handles insurance claims), or issuing an air ticket (i.e., an instance of the process of issuing air tickets) are cases. In this sense, this is analogous to a database instance. Cases are classified in *case types*, i.e., cases that are handled in a similar way. A case has an *identity*, e.g., handling a particular insurance claim is a case that can be univocally identified. A *process* is a procedure followed to handle a particular case type. Processes can be part of other ones, in which case we denote them *sub-processes*. The central component of a workflow is the *task*. A task is a logical, indivisible unit of work. If anything goes wrong when performing a task, it must be rolled-back. This is somehow analogous to the notion of atomicity in transactions in a DBMS.

The notion of *routing* refers to the way in which a process is carried out: it defines the order of the tasks that compose a given process. Routing can be sequential, parallel, selective, or iterative. Finally, tasks can be triggered in different ways: by a resource initiative, by an external event or action (like a message), or by time signals. This is denoted *enactment*. We next study all these concepts in detail, basing ourselves in the classic work by van der Aalst [18].

3.2 Petri Nets

Petri nets, created in 1962 by Carl Adam Petri to model and analyze processes, have been used to model complex processes (particularly in the operating systems field), given their main strength: they are *graphic* tools that have been fully *formalized*. To be used in different areas, basic Petri nets have been extended in many ways. We first present the basic formalism, and then we show how they were extended to model BP. A classic Petri net is a directed bipartite graph defined as follows.

Definition 7 (Petri net). *A Petri net is a triple (P, T, F) where P is a finite set of places, T is a finite set of Transitions, such that $T \cap P = \emptyset$, and $F \subseteq (P \times T) \cup (T \times P)$ is a set of arcs.* \square

A place p is called an *input place* of a transition t if and only if there is an arc from p to t . Conversely, it is called an *output place* if and only if there is

an arc from t to p . Places are represented as circles, and transitions as squares. Directed arcs link both kinds of figures. Each transition has exactly one input place and one output place. Places may contain tokens (probably more than one), indicated as black dots. If a transition takes a token from an input place to put it in an output place, we say that the transition is *fired*. The state of a Petri net is defined by the position of the tokens in it. A transition may only be fired if it is *enabled*, meaning that there is a token in all of its input places.

Example 1 (Petri nets). In Figure 2a we depict a simple Petri net in its initial state. There are three places (A, C, F), and three transitions (B, D, E). There are three tokens in place A. Figure 2b shows that a token has been taken by transition B from place A, and put in place C. Then, there is one process that has been initiated, and two more waiting to be handled.

Graphs containing cycles can model cyclical (i.e., never ending) activities, like traffic lights, for example. \square

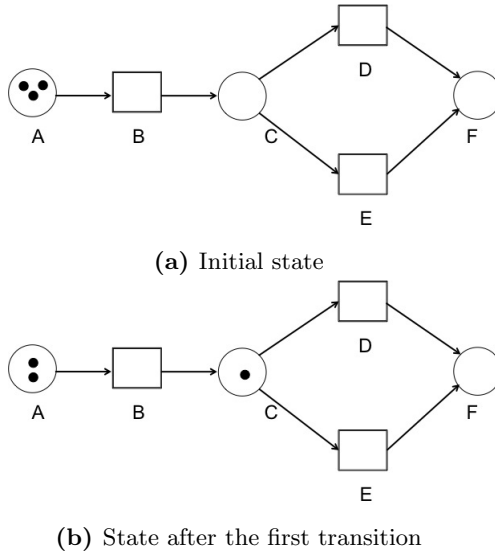


Fig. 2. A Petri Net example

High-Level Petri Nets. Complex processes (like many BP) cannot be expressed in the basic simple Petri net formalism. There are many practical situations which the graph of Definition 7 cannot capture. Thus, this basic definition has been extended in many ways. The three more relevant ones for workflow representation are the *color extension*, the *time extension*, and the *hierarchical extension*. Petri nets extended with color, time and hierarchy are called *high level Petri nets*.

The *color extension* uses colors to identify kinds of tokens. For example, if we are modeling insurance claims, the basic Petri net formalism does not allow to distinguish between a car policy claim and a fire policy claim, which are represented just as tokens. A solution is to assign two different colors for tokens representing each kind of claim. Of course, this is analogous to assigning a value to a token. More interestingly, given that now tokens have values, we could fire transitions based on conditions over these token values. For example “The token to be consumed must correspond to a car insurance code.”

Classic Petri nets do not allow modeling time. Thus, the *time extension* has been defined. In this extension, the token is associated with a timestamp. This way, a token with timestamp ‘10’ can only be consumed from the time instant ‘10’. Before that, the transition is not enabled, even though there is a token in each input place.

To adequately model hierarchical processes, the *hierarchical extension* has been defined. This allows to define sub-processes, represented as a double square indicating that this transition corresponds indeed to a subnet (i.e., another Petri net).

3.3 Representing Workflows with Petri Nets

Now that we have defined the underlying formalism for representing a workflow, let us show how this representation is achieved. For this, each element in a workflow must be mapped to a collection of Petri net constructs. We use the example below to illustrate the discussion.

Example 2 (Workflow¹⁰). An insurance company processes claims regarding car accidents in which its customers are involved. The procedure for processing the insurance claims is the following. Every claim reported by a customer is registered by an employee of the Car Damages (CD) department. After registration, the insurance claim is classified by an employee within CD. There are two categories: *simple and complex* claims. For simple claims, two tasks (*independent from each other*) need to be executed: check insurance and phone garage. Complex claims require three tasks to be executed: check insurance, check damage history, and phone garage. These tasks need to be executed *sequentially in the order specified*. For both, the simple and complex claims, the tasks are performed by employees of department CD. After executing the tasks (simple or complex) a decision is taken, with two possible outcomes: OK (positive) or not_OK (negative). If the decision is positive, the insurance company will pay. An employee of the finance department handles the payment. In any event, the insurance company sends a letter to the customer who submitted the claim. Figure 3 depicts the corresponding workflow diagram. Note that this workflow diagram resembles a Petri net, with some additional notation that we will explain later in detail. □

¹⁰ This example, and some of the figures in this section, have been taken and adapted from <http://wwwis.win.tue.nl/~vvdalst/workflowcourse/>

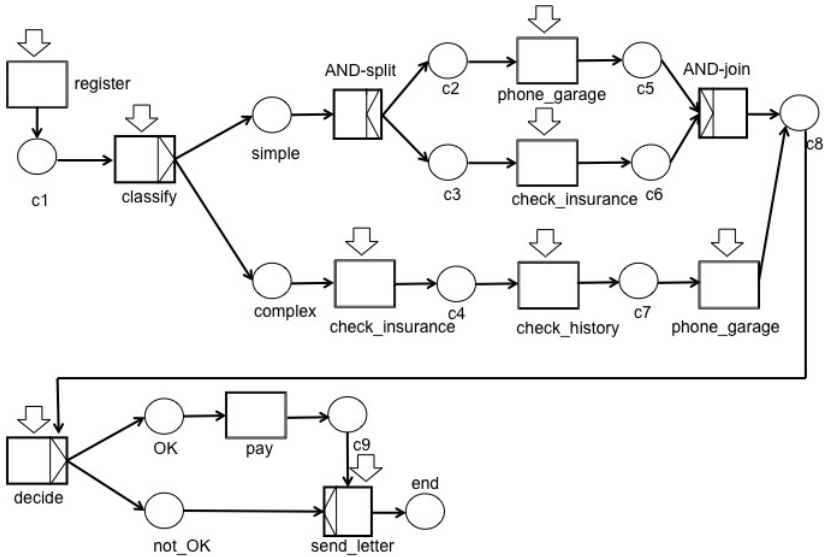


Fig. 3. A workflow example

We next explain how the main elements in a workflow, namely cases, tasks, processes, different case routing schemes, and the different ways of triggering (or enactment) an event, can be represented using Petri nets.

Representing a Process. In Petri nets terminology, the starting point of a process is an *input* place, and the ending point is an *output* place; conditions are represented as places, and tasks as transitions. For example, in Figure 3, the process of registering a claim is represented by the *register* transition. A particular claim starts when this transition is fired, taking a token from the *begin* place and moving it to the place denoted *c1*, where it waits to be classified. The *classify* transition represents a condition; once the transition is fired (in the simplest case, immediately after condition *c1* is satisfied, which means that the transition is enabled), it defines if the case is simple or complex. In the first case, the AND-split transition generates two tokens, initiating two parallel tasks, as we will explain when we address case routing. Figure 3 only shows one token, meaning that at the time depicted by the figure no claims are being processed, and only one case is waiting to be handled. However, in real situations we may have many tokens representing different cases (claims) at different places. In order to distinguish cases from each other, the color extension can be used, assigning different values to tokens corresponding to each cases. Also, tasks in a workflow can be combined in a single process. In this way, a process can be composed of *sub-processes* which, as explained above, can also be represented using the *hierarchy extension* of Petri nets. Figure 4 shows an example with two tokens and a sub-process.

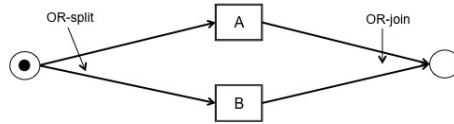


Fig. 6. Non-deterministic implicit OR-split

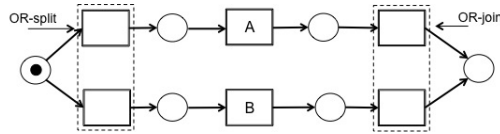


Fig. 7. Non-deterministic explicit OR-split

implicit OR-split, that is, no task represents the decision. When the condition is met, in a non-deterministic way, either task A or task B is executed, but not both. For example, if both tasks are triggered by the occurrence of an event, e.g., the arrival of a kind of document, the first one to get the document will be executed. We can also represent this as an *explicit OR-split*, like in Figure 7. Here, the same artifact as in case (b) is used, i.e., two fictitious tasks (indicated within the dashed square) are added, representing the OR-split. When the condition is satisfied, only one of the ‘fictitious’ tasks is triggered. In this case, although the decision is also non-deterministic, it is taken immediately after the condition is satisfied, which is slightly different than in the implicit case. That means, in the first case, the decision is taken when the tasks must be performed. In the second case, the decision is taken before that, and there is no way back, if the tasks must be switched. Note that in both cases, at the other extreme of the network, an OR-join occurs, and it is explained analogously. Finally, Figure 8 shows a third representation option, a deterministic one. That is, according with the values of the tokens (e.g., using the color extension), a decision is taken. This is also the case of the *classify* task in Figure 3. Note that even when in Figure 8 a special symbol is used, this can also be represented as a Petri net transition that includes a decision rule (i.e., a single square).

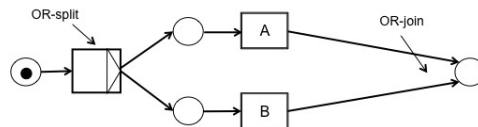


Fig. 8. Deterministic OR-split

Given that AND and OR splits and merges are very frequent in workflows, special symbols are used to represent them, with the meanings explained above. These symbols are depicted in Figure 9.

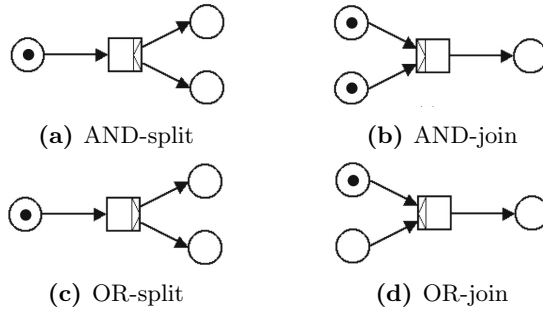


Fig. 9. Representation of split and merge transitions

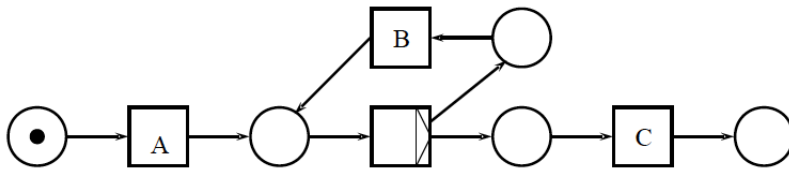


Fig. 10. Iterative routing

Finally, in case (d), *iterative routing* occurs when a task or group of tasks, is executed repeatedly. The typical example is the repetition of a test until the desired outcome is achieved. Figure 10 depicts this situation. Once the task (transition) A is executed (triggered), task B is executed or not, depending on the condition in the OR-split. Note that this corresponds to a While...do kind of loop. The Repeat...until case can also be represented analogously.

Representing Enactment. Figure 3 shows tasks (transitions) annotated by symbols (e.g., an arrow upon task *classify*). Again, these are extensions to the Petri net notation, which allow to represent a wider spectrum of situations in a concise way.



Fig. 11. Types of triggers

Transitions in Petri nets are triggered as soon as the conditions in the places immediately behind them are satisfied. Transitions of this form are represented as single squares (Figure 11a). However, since tasks are associated to cases, we want to represent how, for each case, a task is activated or triggered. We have

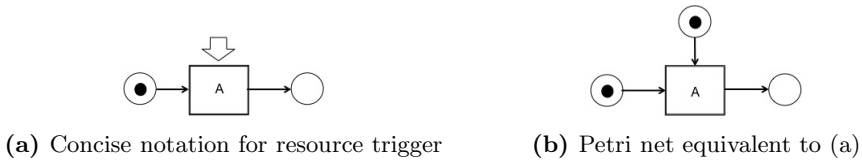


Fig. 12. Equivalence between graphic notations for triggers

typically three choices of triggering: (a) through a resource initiative, like an employee deciding to start the task (Figure 11b); (b) through an external event, like the arrival of a document (Figure 11c); (c) through a time signal, like the definition of an execution time for a task (Figure 11d). Note that this notation is just a shorthand of different high-level Petri nets constructs. For example, for the resource trigger of Figure 12a, Figure 12b shows the classic Petri net equivalent, which requires a ‘trigger’ token. The other cases can be represented analogously.

4 BPMN 2.0

BPMN (Business Process Modeling Notation) provides the tools for defining and understanding the internal and external business procedures, allowing organizations to communicate these procedures in a standard manner. Ideally (we will see that this is not always possible) there should be a mapping from one or more BPMN notation instances to an execution level instance. Thus, BPMN is required to be unambiguous. In a nutshell, the rationale behind BPMN was aimed at: (a) being acceptable and usable by the business community; (b) being constrained to support only the concepts of modeling that are applicable to BP; (c) being useful in describing clearly a complex executable process.

4.1 A Little Bit of History

Figure 13¹¹ summarizes the evolution of workflow management tools, from the early versions until BPMN 2.0 became an OMG standard. The Workflow Management Initiative (WfMI) developed the first process definition language, called Workflow Process Definition Language (WPD), published in 1998. WPD contained all the key concepts required to support workflow automation expressed using URL Encoding. As an evolution of this language, the WfMC developed the XML process Definition Language (XPDL). XPDL was extended to be able to represent in XML all of the concepts present in a BPMN diagram. However, studies showed that the constructs in XPDL do not offer direct support to many of the workflow patterns encountered in practice and present in mature

¹¹ From <http://www.column2.com/2009/05/robert-shapiro-on-bpmn-20/>

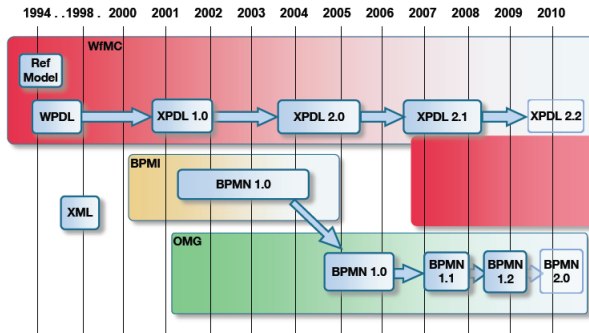


Fig. 13. The BPMN timeline

workflow products. To address this problem, XPD vendors offer specific extensions (see [13] for details).

Originally, BPMN was developed by the BPM Initiative (BPMI). In November, 2002, the BPMN 0.9 draft specification was released to the public, followed by the BPMN 1.0 draft in August 2003. Finally, in May 2004, the BPMN 1.0 specification was released to the public. In 2006 BPMN 1.0 was accepted as an OMG standard. Version 1.2 was accepted in 2008, containing changes in the graphical representation. Actually, many of the constructs explained in Section 3 are present in BPMN, but while the BPMN 1.0 specification did not formally define the semantics of the Business Process Diagram, BPMN 2.0 partially solves this, and also contains significant changes, namely:

- New event types: parallel multiple events.
- Parallel event-based gateway.
- Intermediate events attached to activities.
- Event sub-processes only carried out when an event occurs.
- Updates on collaboration modeling.
- Two new diagram types: (a) *Choreography* diagram, modeling data exchange between partners, where each data exchange is modeled as an activity; (b) *Conversation* diagram, an overview of several partners and their links.

The most relevant changes refer to the way in which diagrams are specified. Former versions only contain informal descriptions of the diagrams. BPMN 2.0 graphic tools are explained using UML. Therefore, a precise semantics is given. An execution semantics is also defined, meaning that the interpretation and execution of BPMN models is precisely described, even including rules to transform BPMN into BPEL format.

4.2 Elements

BPMN is aimed at providing a simple and understandable mechanism for creating BP models, and, at the same time, being able to handle the complexity

inherent to BP. The approach taken to handle these two conflicting requirements was to organize the graphical aspects of the notation into specific categories. This provides a small set of notation categories so that the reader of a BPMN diagram can easily recognize the basic types of elements and understand the diagram. Within the basic categories of elements, additional variation and information can be added to support the requirements for complexity without dramatically changing the basic look-and-feel of the diagram. These five basic categories of elements are:

- Flow Objects. These are the main graphical elements for defining the behavior of a BP. There are three kinds of flow objects; (1) Events; (2) Activities; (3) Gateways. Their basic forms are depicted in Table 1, while more complex forms are depicted in Table 2. These two tables partially reproduce the OMG standard specification¹². Note that the specification *does not state how conditions in gateways must be written*. This is left to the modeler. Also, it must be clear that gateways represent *only logic*, that means, if an activity should take a decision, it should be modeled as a task followed by a gateway. Other modeling options exist, but studying them is out of this paper's scope.
- Data Objects. Data are represented with the following four elements: (1) Data Objects; (2) Data Inputs; (3) Data Outputs; (4) Data Stores.
- Connecting Objects. There are four ways of connecting flow objects to each other, or to other information objects: (1) Sequence Flows; (2) Message Flows; (3) Associations; (4) Data Associations.
- Swimlanes. Used to group the primary modeling elements. Can be of two forms: Pools and Lanes.
- Artifacts. Used to provide additional information about the process. There are two standardized artifacts, but modelers or modeling tools are free to add as many artifacts as necessary. The current set of artifacts includes Group and Text Annotation.

The reader may recognize many of the features studied in Section 3 in Tables 1 and 2. Also the sequence flow semantics is similar, since it is based on tokens: every time a process (a *case*) is started, a start event (Table 2, line 1) creates a token. Then, the token is moved on to the first activity (Table 1, line 2, or any of the complex activities/tasks in Table 2). When the task is performed, the activity passes on the token (or tokens that could have been created). Figure 14 shows a portion of the diagram in Figure 3 (corresponding to Example 2), written in the BPMN formalism. The process starts with a start event (Table 2, line 1), not shown in the figure. The *register* task is followed by a *classify* task. Then, a condition checks if the claim is simple or complex. In the first case, a parallel gateway (which replaces the AND-split) generates two parallel paths, conforming tasks. In this case, connections are sequence flows.

¹² <http://www.bpmn.org/>

Table 1. Basic Elements






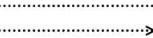





| # | Element | Description | Notation |
|----|-----------------|--|--|
| 1 | Event | Something that happens during the course of a process or a choreography. There are three types of events: start, intermediate, and end. |  |
| 2 | Activity | A work performed during a process. It can be atomic or non-atomic (compound). There are two types of activities in a process model: <i>sub-process</i> and <i>task</i> . |  |
| 3 | Gateway | Used to control the divergence and convergence of sequence flows in a process and in a choreography. |  |
| 4 | Sequence Flow | Shows the order in which activities will be performed in a process and in a choreography. |  |
| 5 | Message Flow | Shows the flow of messages between two participants that are prepared to send and receive them. |  |
| 6 | Association | Links information and artifacts with BPMN graphical elements. |  |
| 7 | Pool | A graphical representation of a participant in a collaboration. It may contain internal details, or it can be a “black box.” |  |
| 8 | Lane | A sub-partition within a process, sometimes within a pool. Extends the entire length of the process, either vertically or horizontally. |  |
| 9 | Data Object | Use to model data associated with processes. Can represent a single object or a collection of objects (see Table 2). |  |
| 10 | Message | Used to depict the contents of a communication between two participants. |  |
| 11 | Text Annotation | A mechanism to provide additional text information for the reader of a BPMN Diagram. |  |

Table 2. Extended Elements



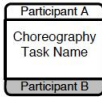

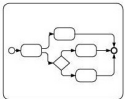
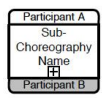
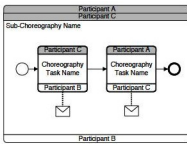








| # | Element | Description | Notation |
|---|----------------------------|--|---|
| 1 | Events | A <i>start</i> event indicates where a particular process or choreography starts. <i>Intermediate</i> events occur between a start event and an end event. The <i>end</i> event indicates where a process or choreography ends. |  |
| 2 | Task (atomic) | An atomic activity in a process. |  |
| 3 | Choreography | An atomic activity in a choreography. It represents a set of one or more message exchange. Each choreography task involves two participants. |  |
| 4 | Collapsed Sub-process | A 'plus' sign in the lower-center of the icon indicates that the activity is a sub-process and has a lower level of detail. |  |
| 5 | Expanded Sub-process | The boundary of the sub-process is expanded and the details (a process) are visible. Note that sequence flows cannot cross the boundary of a sub-process. |  |
| 6 | Collapsed Sub-choreography | The details of the sub-choreography are not visible. A 'plus' sign in the lower-center of the task name band indicates that the activity is a sub-process. |  |
| 7 | Expanded Sub-choreography | The details (a choreography) are visible within the boundary. Sequence flows cannot cross the boundary of a sub-choreography. |  |
| 8 | Gateway types | The types of control include: (a) exclusive decision and merging; (b) event-based and parallel event-based gateways can start a new instance of the process; (c) inclusive gateway decision and merging; (d) complex gateway – complex conditions, e.g., 3 out of 5; (e) parallel gateway forking and joining. Each type of control affects both the incoming and outgoing flow. | <p>Exclusive  or </p> <p>Event-Based  </p> <p>Parallel Event-Based </p> <p>Inclusive </p> <p>Complex </p> <p>Parallel </p> |

Table 2. (continued)

| # | Element | Description | Notation |
|----|--------------------|--|--|
| 9 | Conditional Flow | A sequence flow with a condition evaluated at runtime, to determine whether or not the sequence flow will be used. | |
| 10 | Exception Flow | Occurs outside the normal flow of the process and is based upon an intermediate event that occurs during the performance of the process. | |
| 11 | Data Objects | Can be a singular object or a collection of objects. Besides, a data object can be the input to or the output from a process. | <div style="display: flex; justify-content: space-around;"> <div style="text-align: center;"> Data Object </div> <div style="text-align: center;"> Data Object (collection) </div> </div> <div style="display: flex; justify-content: space-around; margin-top: 5px;"> <div style="text-align: center;"> Data Input </div> <div style="text-align: center;"> Data Output </div> </div> |
| 12 | Fork | Splits a path into two or more parallel ones (AND-Split). Two options: (a) multiple outgoing sequence flows. (b) parallel gateways (normally used in combination with other gateways). | |
| 13 | Join | Combines two or more parallel paths into one path (also denoted AND-Join or synchronization). | |
| 14 | Merging | Combines two or more paths into one path (also denoted OR-Join). If all the incoming flow is alternative, then a gateway is not needed. | |
| 15 | Activity Loop | Tasks and sub-processes performed repeatedly are indicated by a small looping icon at the bottom-center of the activity. | |
| 16 | Sequence Flow Loop | Loops created by connecting a sequence flow to an upstream object. | |

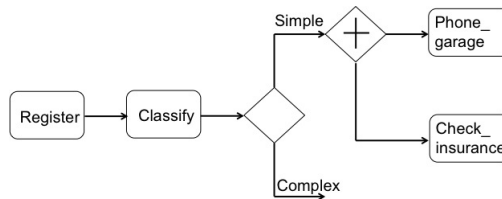


Fig. 14. A portion of Example 2 expressed in BPMN

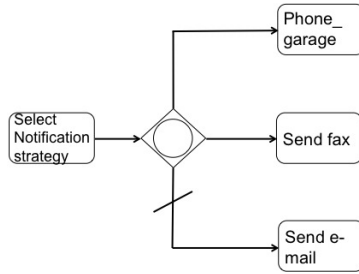


Fig. 15. Inclusive gateway example

We next comment on some BPMN constructs.

Gateways represent splits and merges as studied in Section 3. For example, a parallel gateway (Table 2, line 8) could be used to represent an AND-split (with the same semantics: a token is created for each output path). Analogously, exclusive gateways can model an OR-split. Inclusive gateways allow selecting or merging one or more paths, and are a little more involved. Figure 15 depicts a possible extension to our running example: instead of specifying the `phone_garage` kind of notification, assume that fax or email are now alternative notification options: any possible combination of these three kinds of notification will make the token move. However, in this example we have defined e-mail as the *default flow* (indicated by the line that crosses the flow that goes from the gateway to the activity), meaning that communication via e-mail cannot be chosen in combination with any of the other two. That is, notification can be done just by one of phone, fax, or email, by both phone *and* fax, but *not* via email *and* fax.

Gateways are not the only way for representing splitting and merging. For example, an exclusive gateway splitting into two paths could be replaced by *two conditional flows*, (Table 2, line 9), if the conditions are *mutually exclusive*. Inclusive gateways could be replaced by conditional flows, even when the former constraint does not apply. Moreover, sequence flows could be used to represent parallel gateways. However, BPMN modeling without gateways is not always possible.

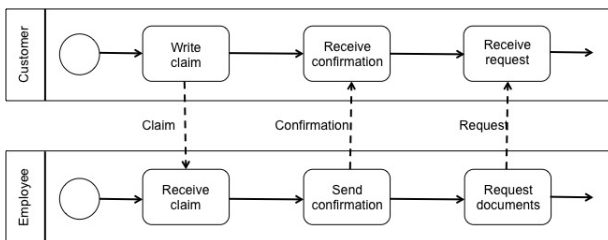


Fig. 16. Collaboration diagram for customer-employee information exchange

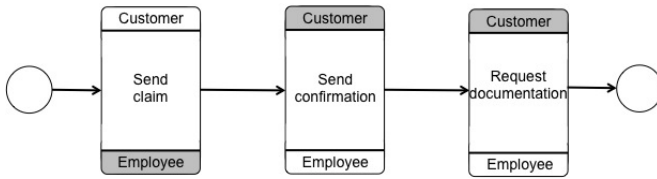


Fig. 17. Choreography diagram for Figure 16

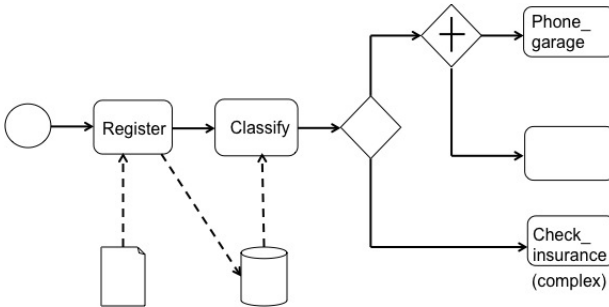


Fig. 18. Representing data objects in the running example

Collaboration is another feature originally present in BPMN, which has been extended with the *Orchestration* feature. Figure 16 shows a collaboration diagram that illustrates the information exchange between a customer sending a claim, and an employee handling it. Flow messages are depicted in dashed lines. The flows are shown in two pools (Table 1, line 7), one for the customer, and one for the employee. This also illustrates that message flows (Table 1, line 5) can only occur between two pools, and *never* within the same pool. *Choreography* diagrams (Table 2, lines 3, 6, and 7) provide another way of representing collaboration. In these diagrams the focus is on the exchange messages themselves, modeled as choreography activities. Figure 17 shows the choreography diagram corresponding to Figure 16. The shadowed band of each exchange box indicates the *receiver* of the message.

When a process uses data, files, or documents, a sequence flow from one activity to another is accompanied by a data transfer. BPMN accounts for this by means of *data objects* (Table 2, line 11). Figure 18 shows how claim documents can be represented in a BPMN graph. A claim document is received, information is stored in a database containing the claims, and this database is read to analyze the type of claim.

5 BPEL: Executing BPM Diagrams

For BP implemented in a language such as Java, their representation as abstract models like the ones commented above is not an easy task. This gap is

slowly being bridged by declarative standards that facilitate the design, deployment, and execution of BP. In particular, the BPEL standard (Business Process Execution Language) [7], short for Web Services-BPEL (WS-BPEL) is an XML-based language to describe the interface between the participants in a process, as well as the full operational logic of the process and its execution flow. BPEL is an OASIS¹³ standard executable language for specifying actions within BP with web services. Processes in BPEL export and import information by using web service interfaces exclusively. Providing a language for the specification of executable and abstract business processes, BPEL extends the web services interaction model and enables it to support business transactions.

BPEL aims at enabling what is called *programming in the large*, a term that refers to the high-level state transition interactions of a process. BPEL refers to this concept as an *abstract process*. A BPEL abstract process represents a set of publicly observable behaviors in a standardized fashion. It includes information such as when to wait for messages, when to send messages, when to compensate for failed transactions, etc. Opposite to this, the term *programming in the small* refers to short-lived programmatic behavior, often executed as a single transaction involving access to local resources such as files, databases, etc. BPEL's development is built on the notion that both kinds of programming require different types of languages. IBM and Microsoft had each defined their own "programming in the large" languages: WSFL and XLANG, respectively. Following the increasing popularity of BPEL, they decided to combine these languages into a new one, denoted BPEL4WS, which allows the formal specification of BP and business interaction protocols by means of extending the web services interaction model, enabling it to support business transactions.

We remark that BPEL is an *orchestration* language, rather than a *choreography* language. An orchestration specifies an executable process that involves message exchanges with other systems, such that the message exchange sequences are controlled by the orchestration designer. Since BPEL adopts web services as its external communication mechanism, its messaging facilities depend on the use of the Web Services Description Language (WSDL) 1.1 to describe outgoing and incoming messages. It should also be clear that BPEL is *not a modeling language* but a *programming language*. This is somehow misunderstood due to the many graphical graphical interfaces and editors that allow for a simple, intuitive design of BPEL specifications associated to vendor products to facilitate code generation. Given that there is no standard graphical notation for BPEL, vendors had created their own notations, taking advantage of the fact that most constructs in BPEL are block-structured (e.g., sequence, while, pick, scope, etc.), enabling a direct visual representation of BPEL process descriptions¹⁴. Some vendors have even proposed to use BPMN as a graphical front-end to capture BPEL process descriptions. As an illustration of the feasibility of this approach, the BPMN

¹³ Organization for the Advancement of Structured Information Standards
<http://www.oasis-open.org/>

¹⁴ See for example Oracle's Jdeveloper interface at
<http://www.oracle.com/technology/bpel/>

specification includes an informal and partial mapping from BPMN to BPEL 1.1. A more detailed mapping of BPMN to BPEL has been implemented in a number of tools, including the open-source BPMN2BPEL¹⁵. However, the development of these tools has revealed fundamental differences between BPMN and BPEL which make it very difficult, and in some cases impossible, to generate readable BPEL code from BPMN models. Even more difficult is the problem of generating BPEL code from BPMN diagrams and maintaining the original BPMN model and the generated BPEL code synchronized, in the sense that any modification to one gets propagated to the other.

In [10] the authors aim at answering the question: can every BPMN model be translated into a BPEL model? They claim that, for a core subset of BPMN which includes parallelism and event-driven choice, the answer is ‘yes’. However, the resulting translation heavily uses a construct in BPEL known as “event handler” which serves to encode event-action rules. Thus, the process model is decomposed into a large number of event-action rules that trigger one another to capture the process flow, and the resulting BPEL code turns out to be unreadable and therefore unsuitable for refinement by developers. The paper also addresses the following question: are there classes of BPMN models that can be translated to BPEL models using the syntactically constrained control flow constructs of BPEL? They identify subsets of BPMN for which this is possible. We refer the interested reader to that paper for details.

5.1 A BPEL Example

In this section we give the flavor of BPEL. For this, we use an example taken from [20]. The example refers to the process of making a reservation of a flight, together with car rental and hotel booking. The process also includes an initial verification of the customer’s credit card. Figure [19] depicts this process in a BPEL4WS graphical representation provided by IBM’s Websphere BPEL editor¹⁶. Figure [20] shows the process expressed in BPMN. Note that there is a *Data Map* activity in the BPMN diagram, which looks rather awkward. Actually, data maps are needed in BPEL to assign variables passed through messages. In the presence of loops, like the one following the *Data Map* activity, the mapping must be explicit. This shows one of the reasons why direct translations from BPMN to BPEL are not always possible: the diagram in Figure [20] was built with the BPEL diagram of Figure [19] in mind, otherwise, the *Data Map* activity would have not been present, since it involves details not needed at a conceptual level.

We now show the BPEL code for portions of this example. Since our goal here is not to give a BPEL tutorial, we omit some language technical details. **Partner link** elements are defined prior to the definition of the process. The *tasks* of the process that are of type **Service**, define the *participants* of the web service. Participants and their properties map to **partner link** elements. Below we show two partner links: the process starter participant, with *business role*

¹⁵ <http://code.google.com/p/bpmn2bpe1/>

¹⁶ <http://publib.boulder.ibm.com/infocenter/adiehelp/v5r1m1/index.jsp?>

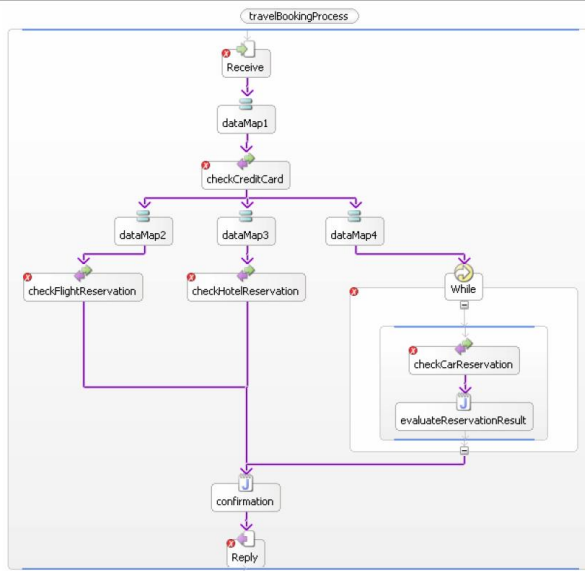


Fig. 19. A BPEL example: a flight reservation process (from [20])

denoted `TravelProcessRole`, and the `HotelReservationService` participant, with role denoted `HotelReservationRole`.

```

<partnerLinks>
  <partnerLink myRole="travelProcessRole"
    name="ProcessStarter"
    partnerLinkType="wsdl5:travelProcess" />
  <partnerLink name="HotelReservationService"
    partnerRole="HotelReservationRole" />
  ...
</partnerLinks>

```

Variables (and their parts) are declared in the BPEL document before defining the process that uses them. They are associated with the properties of a process in a BPMN diagram, and reference `message` elements defined in a WSDL document that supports the BPEL document. Some of the variables and message parts defined for the process of Figure 20 are displayed next.

```

<variables>
  <variable messageType="wsdl0:input" name="input" />
  <variable
    messageType="wsdl4:doCreditCardCheckingRequest"
    name="checkCreditCardRequest" />
  ...
</variables>

```

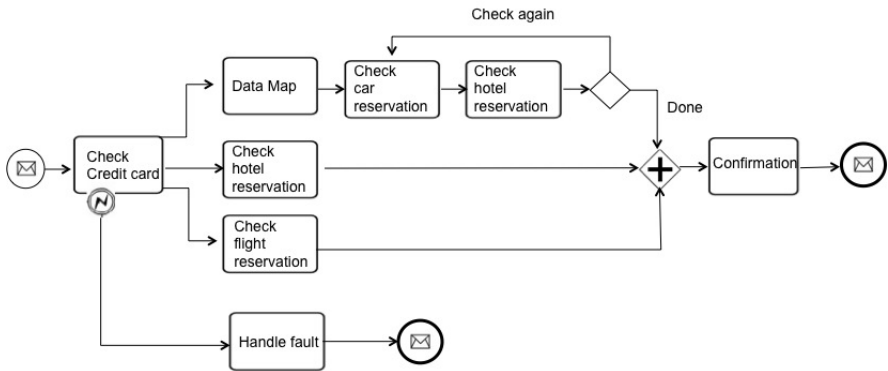


Fig. 20. The BPMN translation of Figure 19

The variable parts of the message are defined in the WSDL document, and look as follows.

```
<message name="input">
  <part name="airline" type="xsd:string"/>
  <part name="arrival" type="xsd:string"/>
  <part name="departure" type="xsd:string"/>
  ...
</message>
```

Sequence flows in Figure 20 map to BPEL link elements. Actually, the BPEL implementation of the BPMN diagram requires more links than the sequence flows shown in Figure 20. This will become clear shortly. The BPEL code for defining the links is:

```
<flow name="Flow" wpc:id="1"/>
  <links>
    <link name="link1"/>
    <link name="link2"/>
    ...
  </links>
</flow>
```

The process starts with a message request for booking a travel itinerary, through a start event called *receive* (Figure 19). This maps to a BPEL *receive* element as shown below. The start event in the BPMN diagram of Figure 20 is implemented as a web service by the process starter participant, and receives the message with name 'input'. The source link *link1* represents the sequence flow between the *receive* event and the data map.


```

<receive createInstance="yes" operation="book"
  name="Receive"
  wpc:displayName="Receive"
  portType="wsdl0:travelPort"
  variable="input" wpc:id="2">
  <source linkName="link1" />
</receive>

```

After the request has been received, the validity of the credit card information submitted is checked. This is represented by the BPMN activity *Check Credit Card*, and the BPEL task *checkCreditCard*. (Note the error intermediate event -Table 2 line 1- in the BPMN diagram; this event is used for handling an incorrect credit card number). This task requires a data mapping (indicated by '=' in Figure 19), represented as a BPEL `assign` element. The data mapping implements the assignment of the data received from the input to the data structure of the message that will handle the credit card checking service. Note that for the *Check Flight Reservation* and *Check Hotel Reservation* activities, this data assignment was considered as part of the activity, while for the *Check Car Reservation* activity, involved in a loop, this is not possible, producing a mismatch between the BPMN and BPEL representations. Even though the data mapping as a group is mapped to a BPEL `assign` element, individual property mappings are mapped to `copy` elements within the latter. The code below illustrates this discussion.

```

<assign name="dataMap1" wpc:displayName="dataMap1"
  wpc:id="20">
  <target linkName="link1"/>
  <source linkName="link2"/>
  <copy>
    <from part="cardNumber" variable="input"/>
    <to part="cardNumber"
      variable="checkCreditCardRequest"/>
  </copy>
  <copy>
    <from part="cardType" variable="input"/>
    <to part="cardType"
      variable="checkCreditCardRequest"/>
  </copy>
</assign>

```

The above code represents the data mapping `dataMap1` in Figure 19. In that figure, the flow between `dataMap1` and `checkCreditCard` is mapped to the link `link2`, represented by the `source` element within the `assign` element `dataMap1`. Data from the `input` variable, is copied to data in the `checkCreditCardRequest` variable, which is the input variable in the next task, namely, *checkCreditCard*. This task is implemented as a web service, and represented, as we explained before, by an `invoke` element. (we recall that

both, the data map - an `assign` element - and `checkCreditCard` - `invoke` element - are represented by the single activity *Check Credit Card* in the BPMN diagram. Here, the participant is the `CreditCardCheckingService` partner link, through the `wsdl4:creditCardCheckingServiceImpl` interface and the `doCreditCardChecking` operation. The output variable is denoted `checkCreditCardResponse`. We show this task next.

```
<invoke inputVariable="checkCreditCardRequest"
  name="checkCreditCard"
  operation="doCreditCardChecking"
  outputVariable="checkCreditCardResponse"
  partnerLink="CreditCardCheckingService"
  portType="wsdl4:CreditCardCheckingServiceImpl"
  wpc:displayName="Check Credit Card" wpc:id="5">
  <target linkName="link2"/>
  <source linkName="link3"/>
  <source linkName="link6"/>
  <source linkName="link9"/>
</invoke>
```

After the *Check Credit Card* task, three main parallel activities occur, involving the checking of car, hotel, and flight reservations (we do not show the checking of the car reservation). There are also three data mappings before these three tasks. Parallelism is indicated by the three outgoing sequence flow arcs from the *Check Credit Card* task in the BPMN diagram. Each flow is represented by the three BPEL link elements (`link3`, `link6`, and `link9`), outgoing from the previous task (in the code above we can see that the three corresponding link elements are included as `source` elements in the `checkCreditCard` `invoke` element. As before, the mapping of the *Check Flight Reservation* activity results in an `assign` element that precedes an `invoke` element. A link element (`link4`) that does not have a corresponding Sequence Flow in the BPMN diagram must be added to create the sequential dependency between the `assign` and the `invoke` elements. The mapping of the *Check Hotel Reservation* task is similar, involving the creation of a link element (`link4`). We show the BPEL code next.

```
<assign name="DataMap2" wpc:displayName="DataMap2"
  wpc:id="21">
  <target linkName="link3"/>
  <source linkName="link4"/>
  <copy>
    <from part="airline" variable="input"/>
    <to part="airline"
      variable="flightReservationRequest"/>
  </copy>
  ...
</assign>
```

```

<invoke inputVariable="flightReservationRequest"
  name="checkFlightReservation"
  operation="doFlightReservation"
  outputVariable="flightReservationResponse"
  partnerLink="FlightReservationService"
  portType="wsdl3:FlightReservationServiceImpl"
  wpc:displayName="Check Flight Reservation"
  wpc:id="10">
  <target linkName="link4"/>
  <source linkName="link5"/>
</invoke>

```

Finally, we remark again that the *Data Map* activity in the BPMN diagram, preceding the *Check Car Reservation* task, is made explicit (opposite to the *Check Flight Reservation* and *Check Hotel Reservation* activities, where it was included in the tasks' code). This is because the *Check Car Reservation* task is within a loop, although the data mapping is needed only once. Therefore, it must be represented as a separate activity. Below, we show the data mapping in BPEL.

```

<assign name="DataMap4" wpc:displayName="Data Map"
  wpc:id="23">
  <target linkName="link9"/>
  <source linkName="link10"/>
  <copy>
    <from part="carCompany" variable="input"/>
    <to part="company" variable="carReservationRequest"/>
  </copy>
  ...
</assign>

```

We refer the interested reader to [20] for the complete example.

Finally, we remark that there is an increasing number of tools that generate executable business processes generating code directly from BPMN, without using BPEL [17]. Examples of this are jBPM5 [18], Activiti [19], and Roubro [20].

6 Other Workflow Modeling Tools: UML and YAWL

UML. The Unified Modeling Language (UML) [21] is a standard for software development. It has evolved from version 1.0 in 1997 to version 2.4 in 2011.

¹⁷ <http://www.bpm.com/bpel-who-needs-it.html>

¹⁸ <http://www.jboss.org/jbpm/>

¹⁹ <http://activiti.org/>

²⁰ <http://activiti.org/>

²¹ <http://www.omg.org/spec/UML/2.4.1/>

During this period, formalization and semantics have been added. It currently defines fourteen diagrams, seven devoted to structural characteristics, and seven to behavioral ones. Dealing with workflows, our interest is on behavioral diagrams, namely: Activity, Communication (Collaboration in versions 1.x), Interaction, Sequence, State, Timing, and Use Case diagrams. In UML 2.x, the semantics of *Activity diagrams* changed from being based on state-machine semantics to Petri net semantics, expanding the number of cases they can capture. As a consequence, Activity diagrams are now generally more used than state machine diagrams. They are used to model the control flow between objects, being thus a form of flowchart. *Activity diagrams* consist in a collection of shapes, connected by arrows, which run from the start towards the end, representing the order in which activities occur. The most used shapes are rounded rectangles representing activities, diamonds representing decisions, bars representing the starting (split) or ending (join) of concurrent activities, a black circle representing the starting state of the workflow, and an encircled black circle representing the end state. Although aimed at expressing concurrency, the join and split symbols in Activity diagrams only solve simple cases, since the meaning of the model is not clear when they are arbitrarily combined with decisions or loops. *Sequence diagrams* show the order of message exchange between actors in a system, but in their more used form, they do not support choice, synchronization, or iteration. *Communication diagrams* (a simplified version of Collaboration diagrams), depict the organization of objects that participate in an interaction, and, in some way, are similar to Communication diagrams. *State diagrams* are an extension of state machines, being a collection of states and transitions from one state to another.

UML and Petri Nets. From the description above, the relationship between UML and the workflow definitions based on Petri nets, given in Section 3 appears straightforward. A given Activity diagram could be translated into a Petri net mapping activities to transitions, object flows to places, and synchronization bars to transitions. Probably, additional places can be required to connect transitions. Also, a subset of Petri nets, Workflow nets (Petri nets with a unique start place, a unique end place, and where each place is in a path from start to end) can be, in general, translated to an Activity diagram.

UML Activity Diagrams and BPMN. One difference between BPMN and UML Activity diagrams resides in the fact that the latter are, from the start, an execution-oriented language, while, on the other hand, BPMN has been designed with the aim of being a notation for high-level modeling, and BPMN models were not originally intended to be directly executed. Note however, that since BPMN 2.0 has a much more detailed semantics than its predecessors, executable processes can now be built starting from a BPMN model (some examples were given above). The Workflow Patterns framework²² provides a reference analysis framework consisting in a number of patterns which provide a taxonomy of generic, recurring concepts and constructs relevant for process representation and modeling. A comparison between BPMN and UML Activity diagrams in terms of

²² <http://www.workflowpatterns.com>

Workflow Patterns can be found in [21]. The authors report that BPMN provides support for the majority of the control-flow patterns, for nearly half of the data patterns, and only very limited support for a resource perspective (e.g., resource allocation, distribution, handling). They also observe that detailed knowledge of the non-graphically represented attributes of the modelling constructs in BPMN is required in order to solve some of the patterns. Providing a rich graphical notation and support for an extensive set of non-graphical elements leads to an increased complexity. The conclusion is that (in a similar fashion as what we stated above), from a control-flow perspective (which is relevant to BPM), BPMN and UML 2.0 Activity diagrams are almost equivalent, although BPMN is slightly stronger when it comes to the representation of parallel routing and synchronized merge patterns, because of the larger number of control-flow constructs offered by BPMN with respect to UML Activity diagrams, in particular AND/OR-splits, AND/OR-joins, and complex gateways.

As a final comment, it is worth noting that Activity diagrams, although part of the UML OMG standard, have not been widely adopted by the BP community, and their use is basically limited to software development specification.

Yet Another Workflow Language (YAWL). YAWL [15] is a workflow language based on workflow patterns. The language is supported by a software system that includes an execution engine, a graphical editor and a worklist handler. The system is available as open source software under the LGPL license²³. It has been used in many real-world implementations, and extensively for university teaching. YAWL aims at being a workflow language for supporting most of the workflow patterns, while having an underlying formal semantics. Petri nets are the basis of YAWL, which extends them with three main constructs, namely OR-join, cancellation sets, and multi-instance activities. Even though Petri nets (including high level Petri nets) support a number of the identified patterns, they do not provide direct support for the cancellation patterns (in particular the cancellation of a whole case), the synchronizing merge pattern (where all active threads need to be merged, and branches which cannot become active need to be ignored), and patterns dealing with multiple active instances of the same activity in the same case. This motivated the development of YAWL, which combines the insights gained from the workflow patterns with the benefits of Petri nets. In fact, the semantics of YAWL is not defined in terms of Petri nets but rather in terms of a transition system. This was motivated by the fact that some of the extensions to Petri nets were difficult or even impossible to re-encode back into plain Petri nets. The fact that YAWL is based on a formal semantics has enabled the implementation of several techniques for analyzing YAWL processes. In particular, the YAWL system includes a static analysis tool called WofYAWL. It is out of the scope of this work to review the constructs of YAWL, since most of the functionalities have been studied in Section 3.

Regarding execution, YAWL is sometimes considered an alternative to BPEL. A crucial advantage of BPEL is that it is driven by a standardization committee,

²³ <http://www.yawlfoundation.org>

and supported by several IT industry players. Thus, unlike YAWL, there are numerous tools (proprietary and open-source) that support BPEL. Also, several researchers have captured the formal semantics of subsets of BPEL in terms of various formalisms, including Petri nets, process algebra and finite state machines, leading to the development of static analysis tools for BPEL that can compete with the static analysis capabilities provided by the YAWL system.

7 Process Mining

Business intelligence (BI) tools use event data to support decision-making. Within such machinery, data mining (a collection of techniques and algorithms aimed at discovering interesting information in large databases [8]) has been steadily gaining attention, and many BI tools today offer mature data mining capabilities. However, these tools are data-centric rather than process-centric, since they focus on data and local decision making and not in end-to-end processes. In contrast, BPM tools use process models to analyze operational processes. These models are often disconnected from actual event data. Therefore, results tend to be unreliable because they are based on an idealized model of reality and not on observed facts. Then, we are in a situation where business process analysis tools are process-based and data mining tools used to analyze processes are data-based, and almost ignore business processes that data support. *Process mining* [12], a term recently coined, aims to bridge the gap between BI and BPM by combining event data and process models. Unlike traditional approaches, its goal is not to construct a single static model, but rather to dynamically map processes using the most recent data to make predictions or answer particular questions. Process mining joins ideas of process modeling and analysis on the one hand and data mining and machine learning on the other. The industry and academia interest in process mining led the IEEE to establish the Task Force on Process Mining, within the context of the Data Mining Technical Committee (DMTC) of the Computational Intelligence Society (CIS)²⁴. The remainder is based on the process mining manifesto²⁵ issued by this task force, and on [14].

The idea of process mining is to discover, monitor and improve processes by extracting knowledge from event logs available in information systems. Process mining includes (automated) process discovery (i.e., extracting process models from an event log), conformance checking (i.e., monitoring deviations by comparing model and log), organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations.

7.1 The Process

The starting point for process mining is an *event log*. Each event in such a log refers to an activity and is related to a particular *case*. We have already studied

²⁴ <http://www.win.tue.nl/ieetfpm/doku.php>

²⁵ http://www.win.tue.nl/ieetfpm/doku.php?id=shared:process_mining_manifesto

the meaning of these terms, and know that events belonging to a case are ordered and describe one ‘run’ of the process. Event logs can store additional data about events. In fact, whenever possible, process mining techniques use supplementary information such as the resource (person or device) executing or initiating the activity, the events timestamp, and other data attributes such as order size. The mining process has, basically, three stages:

- *Process Discovery*. First, analysts use process discovery techniques to extrapolate a model from an event log. They can also create this initial process model manually. Basically, a discovery technique takes an event log and produces a model without using any apriori information. Well-known data mining techniques are applied to discover processes based on example executions in event logs.
- *Conformance Checking*. After the discovery step, analysts apply conformance checking techniques to diagnose deviations between the event log and the initial process model. This task is carried out comparing an existing process with an event log of the same process. Conformance checking can be used to verify if the processes actually run, and recorded in the log, conform to the model (and vice versa). Note that different types of models can be considered: conformance checking can be applied to procedural models, organizational models, etc.
- *Model Enhancement*. Finally, during model enhancement, analysts use information from the log to repair or extend the model. For example, they can use time stamps to add timing information (waiting times and service times) to the model. The resulting enhanced process model can support decision making. In other words, the idea is to extend or improve an existing process model using information about the actual process recorded in some event log. While conformance checking measures the alignment between model and reality, this third type of process mining aims at changing or extending the model. For instance, by using timestamps in the event log one can extend the model to show bottlenecks, service levels, throughput times, and frequencies.

In summary: process *discovery* receives an event log and returns a model; *conformance checking* receives an event log and a model, and produces a diagnostics; *enhancement* receives a model and an event log, and produces a new model.

Process mining covers different perspectives. The *control-flow perspective* focuses on control flow, i.e., the ordering of activities. Mining from this perspective is aimed at finding a good characterization of all possible paths. The result is typically expressed in terms of a Petri net or some other process notation (e.g., BPMN, or UML Activity diagrams). The *organizational perspective* focuses on information about resources hidden in the log, i.e., which actors (e.g., people, systems, roles, or departments) are involved and how are they related. The goal is to either structure the organization by classifying people in terms of roles and organizational units, or to show the social network. The *case perspective* focuses on properties of cases. Obviously, a case can be characterized by its path in the process or by the actors working on it. However, cases can also be characterized

by the values of the corresponding data elements. For example, if a case represents a replenishment order, it may be interesting to know the supplier or the number of products ordered. The *time perspective* is concerned with the timing and frequency of events. When events bear timestamps it is possible to discover bottlenecks, measure service levels, monitor the utilization of resources, and predict the remaining processing time of running cases. These perspectives can be mined using classic data mining tools.

8 Open Research Problems

In recent years, research on database management attention has been increasingly accounting for the context in which data are generated and manipulated, namely the processes, the users, and the goals that these data serves [5]. Consider the ticket reservation example of Section 5. Research that focuses only on data storage and manipulation can tell us how to design the database, and how to query it in an optimal way. However, in the perspective of the company that runs the ticket and hotel bookings, the database is only a tool used in the company BP, which to the company, is probably as important as the data. Much of the success of database systems is due to the elegance of the relational model and its declarative query languages, combined with a rich spectrum of underlying optimization techniques and efficient implementations. In the context of BP, this is still an open issue. Elegant formal models and query languages are still to be developed. We have studied in this paper many good models and techniques for capturing and analyzing the BP flow, and for capturing and analyzing the data they manipulate. But from our study it also comes clear that data are only considered in a limited way. A comprehensive solution for the explicit modeling and analyzing the processes flow and data, and their interactions is still to be produced. In this context, many research problems appear in the field of BPM. Below we summarize a few ones.

Modeling. One difficulty that arises when attempting an effective management of BP is the typical complexity of their representations. BP usually operate in a cross-organizational, distributed environment and the software implementing them is fairly complex. Like in the case of data management, effective BP management needs to rely on an abstract model for the BP. To answer this need, many different abstract representation models have been suggested, some of them commented in this paper. These models combine, to some extent, flow models with models that describe the underlying databases of the application and their interaction with flow. The biggest research challenge in BPM is the combination of these rich flow model with another rich model for underlying database manipulation. High complexity or even undecidability of analysis is difficult to avoid whenever such rich models are considered. Abiteboul et al. [1] have recently addressed this problem by designing an artifact model for Active XML with an underlying, explicitly modeled, Finite State Machines. Cohn and Hull [4] state that no model is likely to be the best for all needs, and consequently that there

is a need for the development of a theory (and practical implementations) of views on BP and a practical mapping between them.

Process Mining. We addressed this topic in Section 7. Being a new field, it abounds in open research challenges. Some of them are: (a) finding, merging, and cleaning event data; (b) dealing with complex event logs of different characteristics; (c) creating representative benchmarks; (d) cross-organizational mining; (e) providing operational support; (i) combining process mining with other types of analysis; (j) improving usability for non-experts.

Querying. BP are an important source of information for the process owners as well as their users. At least two kinds of analysis are of interest in this context: (a) the analysis of possible future executions of the process. This can be used to pre-empt possible bugs or breach of policies in future executions; (b) querying logs of past executions. To support these kinds of analysis, we need a *formal query language*. Some proposals in this sense exist, like BPQL [2]. However, further study of the trade-off between expressiveness and complexity of query evaluation in this context is a possible research direction. Also, the development of dedicated optimization techniques for the analysis of past, present and future executions is an important challenge.

9 Conclusion

In this paper we have reviewed basic concepts on business process management, mainly related to process modeling. We started from the notion of workflow management and Petri nets. With this background, we discussed the BPMN 2.0 standard and compared it with other process modeling techniques, like UML activity diagrams, and YAWL. We addressed business process execution, through an overview of BPEL. We concluded with a brief account of the emerging area of process mining, and an analysis of open research directions.

References

1. Abiteboul, S., Bourhis, P., Vianu, V.: Comparing workflow specification languages: a matter of views. In: 14th International Conference on Database Theory-ICDT 2011, pp. 78–89 (2011)
2. Beeri, C., Eyal, A., Milo, T., Pilberg, A.: Monitoring business processes with queries. In: Proceedings of the 33rd International Conference on Very Large Data Bases, VLDB, pp. 603–614 (2007)
3. Workflow Management Coalition. Terminology and glossary. Document Number WFMC-TC-1011 3.0 (February 1999)
4. Cohn, D., Hull, R.: Business artifacts: A data-centric approach to modeling business operations and processes. IEEE Data Engineering Bulletin 32(3), 3–9 (2009)
5. Deutch, D., Milo, T.: A quest for beauty and wealth (or, business processes for database researchers). In: Proceedings of the 30th ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2011, pp. 1–12 (2011)

6. Ellis, C.A.: Information control nets: A mathematical model of office information flow. In: Proceedings of the Conference on Simulation, Measurement and Modeling of Computer Systems, vol. 8(3), pp. 225–240. ACM (1979)
7. Organization for the Advancement of Structured Information Standards (OASIS). WSPeL 2.0 (April 2007), <http://www.oasis-open.org/standards#wsbpelv2.0>
8. Han, J., Kamber, M.: Data mining: concepts and techniques. The Morgan Kaufmann series in data management systems. Elsevier (2006)
9. Holt, A.: Coordination Technology and Petri Nets. In: Rozenberg, G. (ed.) APN 1985. LNCS, vol. 222, pp. 278–296. Springer, Heidelberg (1986)
10. Ouyang, C., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M.: From business process models to process-oriented software systems: The BPMN to BPEL way (October 2006)
11. Peterson, J.L.: Petri Net Theory and the Modeling of Systems. Prentice-Hall (1981)
12. van der Aalst, W.M.P.: Using process mining to bridge the gap between BI and BPM. *Computer* 44(12), 77–80 (2011)
13. van der Aalst, W.M.P.: Patterns and XPD L: A critical evaluation of the XML process definition language. Technical report FIT-TR-2003-06, Queensland University of Technology, Brisbane, Australia (2003)
14. van der Aalst, W.M.P.: Process Mining. Springer (2011)
15. van der Aalst, W.M.P., Alred, L., Dumas, M., ter Hofstede, A.H.M.: Design and implementation of the YAWL system. Technical report FIT-TR-2003-07, Queensland University of Technology, Brisbane, Australia (2003)
16. van der Aalst, W.M.P., ter Hofstede, A.H.M.: YAWL: yet another workflow language. *Information Systems* 30(4), 245–275 (2005)
17. van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M.: Business Process Management: A Survey. In: van der Aalst, W.M.P., ter Hofstede, A.H.M., Weske, M. (eds.) BPM 2003. LNCS, vol. 2678, pp. 1–12. Springer, Heidelberg (2003)
18. van der Aalst, W.M.P., van Hee, K.M.: Workflow Management: Models, Methods, and Systems. MIT Press (2002)
19. van der Aalst, W.M.P.: Three good reasons for using a Petri-net-based workflow management system. In: Proceedings of the International Working Conference on Information and Process Integration in Enterprises (IPIC 1996), Cambridge, MA, pp. 179–201 (1996)
20. White, S.: Using BPMN to model a BPEL process. *BPTrends* 3(3), 1–18 (2005)
21. Wohed, P., van der Aalst, W.M.P., Dumas, M., ter Hofstede, A.H.M., Russell, N.: Pattern-based analysis of BPMN (2005), <http://eprints.qut.edu.au/2977/>

Machine Learning Strategies for Time Series Forecasting

Gianluca Bontempi, Souhaib Ben Taieb, and Yann-Aël Le Borgne

Machine Learning Group
Computer Science Department, Faculty of Sciences
ULB, Université Libre de Bruxelles
Bd Triomphe, 1050, Brussels, Belgium
{gbonte, sbentaieb, yleborgn}@ulb.ac.be
<http://mlg.ulb.ac.be>

Abstract. The increasing availability of large amounts of historical data and the need of performing accurate forecasting of future behavior in several scientific and applied domains demands the definition of robust and efficient techniques able to infer from observations the stochastic dependency between past and future. The forecasting domain has been influenced, from the 1960s on, by linear statistical methods such as ARIMA models. More recently, machine learning models have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community. This chapter presents an overview of machine learning techniques in time series forecasting by focusing on three aspects: the formalization of one-step forecasting problems as supervised learning tasks, the discussion of local learning techniques as an effective tool for dealing with temporal data and the role of the forecasting strategy when we move from one-step to multiple-step forecasting.

Keywords: Time series forecasting, machine learning, local learning, lazy learning, MIMO.

1 Introduction

A *time series* is a sequence S of historical measurements y_t of an observable variable y at equal time intervals. Time series are studied for several purposes such as the forecasting of the future based on knowledge of the past, the understanding of the phenomenon underlying the measures, or simply a succinct description of the salient features of the series. In this chapter we shall confine ourselves to the problem of forecasting. Forecasting future values of an observed time series plays an important role in nearly all fields of science and engineering, such as economics, finance, business intelligence, meteorology and telecommunication [43]. An important aspect of the forecasting task is represented by the size of the horizon. If the one-step forecasting of a time series is already a challenging task, performing multi-step forecasting is more difficult [53] because of

additional complications, like accumulation of errors, reduced accuracy, and increased uncertainty [58,49].

The forecasting domain has been influenced, for a long time, by linear statistical methods such as ARIMA models. However, in the late 1970s and early 1980s, it became increasingly clear that linear models are not adapted to many real applications [25]. In the same period, several useful nonlinear time series models were proposed such as the bilinear model [44], the threshold autoregressive model [56,54,55] and the autoregressive conditional heteroscedastic (ARCH) model [22] (see [25] and [26] for a review). However, the analytical study of nonlinear time series analysis and forecasting is still in its infancy compared to linear time series [25].

In the last two decades, machine learning models have drawn attention and have established themselves as serious contenders to classical statistical models in the forecasting community [143,61]. These models, also called black-box or data-driven models [40], are examples of nonparametric nonlinear models which use only historical data to learn the stochastic dependency between the past and the future. For instance, Werbos found that Artificial Neural Networks (ANNs) outperform the classical statistical methods such as linear regression and Box-Jenkins approaches [59,60]. A similar study has been conducted by Lapedes and Farber [33] who conclude that ANNs can be successfully used for modeling and forecasting nonlinear time series. Later, other models appeared such as decision trees, support vector machines and nearest neighbor regression [29,3]. Moreover, the empirical accuracy of several machine learning models has been explored in a number of forecasting competitions under different data conditions (e.g. the NN3, NN5, and the annual ESTSP competitions [19,20,34,35]) creating interesting scientific debates in the area of data mining and forecasting [28,45,21].

This chapter aims to present an overview of the role of machine learning techniques in time series forecasting by focusing on three aspects: the formalization of one-step forecasting problems as supervised learning tasks, the discussion of local learning techniques as an effective tool for dealing with temporal data and the role of the forecasting strategy when we move from one-step to multi-step forecasting.

The outline of the chapter is as follows. Section 2 introduces some basic notions of time series modeling and the formalization of the forecasting task as an input-output problem. Section 3 discusses the role of machine learning techniques in inferring accurate predictors from observed data and introduces the local learning paradigm. Section 4 presents several strategies for multi-step forecasting which have been proposed so far in literature. Section 5 reviews how local learning techniques have been integrated with multiple-step strategies to perform accurate multi-step forecasts.

2 Forecasting and Modeling

Two main interpretations of the forecasting problem on the basis of historical dataset exist. Statistical forecasting theory assumes that an observed sequence

is a specific realization of a random process, where the randomness arises from many independent degrees of freedom interacting linearly [4]. However, the emergent view in dynamical systems theory [23,17] is that apparently random behavior may be generated by deterministic systems with only a small number of degrees of freedom, interacting nonlinearly. This complicated and aperiodic behavior is also called *deterministic chaos* [48].

We adopt the working hypothesis that many classes of experimental time series may be analyzed within the framework of a dynamical systems approach. Therefore the time series is interpreted as the observable of a dynamical system whose state s evolves in a state space $\Gamma \subset \mathbb{R}^g$, according to the law

$$s(t) = \mathcal{F}^t(s(0)) \quad (1)$$

where $\mathcal{F} : \Gamma \rightarrow \Gamma$ is the map representing the dynamics, \mathcal{F}^t is its iterated versions and $s(t) \in \Gamma$ denotes the value of the state at time t .

In the absence of noise the time series is related to the dynamical system by the relation

$$y_t = \mathcal{G}(s(t)) \quad (2)$$

where $\mathcal{G} : \Gamma \rightarrow \mathbb{R}^D$ is called the *measurement function* and D is the dimension of the series. In the following we will restrict to the case $D = 1$ (*univariate time series*).

Both the function \mathcal{F} and \mathcal{G} are unknown, so in general we cannot hope to reconstruct the state in its original form. However, we may be able to recreate a state space that is in some sense equivalent to the original.

The *state space reconstruction problem* consists in reconstructing the state when the only available information is contained in the sequence of observations y_t . State space reconstruction was introduced into dynamical systems theory independently by Packard *et al.* [42] and Takens [52]. The Takens theorem implies that for a wide class of deterministic systems, there exists a mapping (*delay reconstruction map*) $\Phi : \Gamma \rightarrow \mathbb{R}^n$

$$\Phi(s(t)) = \{\mathcal{G}(\mathcal{F}^{-d}(s(t))), \dots, \mathcal{G}(\mathcal{F}^{-d-n+1}(s(t)))\} = \{y_{t-d}, \dots, y_{t-d-n+1}\} \quad (3)$$

between a finite window of the time series $\{y_{t-d}, \dots, y_{t-d-n+1}\}$ (*embedding vector*) and the state of the dynamic system underlying the series, where d is called the *lag time* and n (*order*) is the number of past values taken into consideration. Takens showed that generically Φ is an *embedding* when $n \geq 2g + 1$, where embedding stays for a smooth one-to-one differential mapping with a smooth inverse [17]. The main consequence is that, if Φ is an embedding then a smooth dynamics $f : \mathbb{R}^n \rightarrow \mathbb{R}^n$ is induced in the space of reconstructed vectors

$$y_t = f(y_{t-d}, y_{t-d-1}, \dots, y_{t-d-n+1}) \quad (4)$$

This implies that the reconstructed states can be used to estimate f and consequently f can be used in alternative to \mathcal{F} and \mathcal{G} , for any purpose concerning time series analysis, qualitative description, forecasting, etc.

The representation (4) does not take into account any noise component, since it assumes that a deterministic process f can accurately describe the time series. Note, however, that this is simply one possible way of representing the time series phenomenon and that any alternative representation should not be discarded a priori. In fact, once we assume that we have not access to an accurate model of the function f , it is perfectly reasonable to extend the deterministic formulation (4) to a statistical Nonlinear Auto Regressive (NAR) formulation

$$y_t = f(y_{t-d}, y_{t-d-1}, \dots, y_{t-d-n+1}) + w(t) \quad (5)$$

where the missing information is lumped into a noise term w . In the rest of the chapter, we will then refer to the formulation (5) as a general representation of the time series which includes as particular instance also the case (4).

The success of a reconstruction approach starting from a set of observed data depends on the choice of the hypothesis that approximates f , the choice of the order n and the lag time d .

In the following section we will address only the problem of the modeling of f , assuming that the values of n and d are available a priori. A good reference on the order selection is given in Casdagli *et al.* [17].

3 Machine Learning Approaches to Model Time Dependencies

3.1 Supervised Learning Setting

The embedding formulation in (5) suggests that, once a historical record S is available, the problem of one-step forecasting can be tackled as a problem of supervised learning. Supervised learning consists in modeling, on the basis of a finite set of observations, the relation between a set of *input* variables and one or more *output* variables, which are considered somewhat dependent on the inputs. Once a model of the mapping (5) is available, it can be used for one-step forecasting. In one-step forecasting, the n previous values of the series are available and the forecasting problem can be cast in the form of a generic regression problem as shown in Fig. 1.

The general approach to model an input/output phenomenon, with a scalar output and a vectorial input, relies on the availability of a collection of observed pairs typically referred to as *training set*.

In the forecasting setting, the training set is derived by the historical series S by creating the $[(N - n - 1) \times n]$ input data matrix

$$X = \begin{bmatrix} y_{N-1} & y_{N-2} & \dots & y_{N-n-1} \\ y_{N-2} & y_{N-3} & \dots & y_{N-n-2} \\ \vdots & \vdots & \vdots & \vdots \\ y_n & y_{n-1} & \dots & y_1 \end{bmatrix} \quad (6)$$

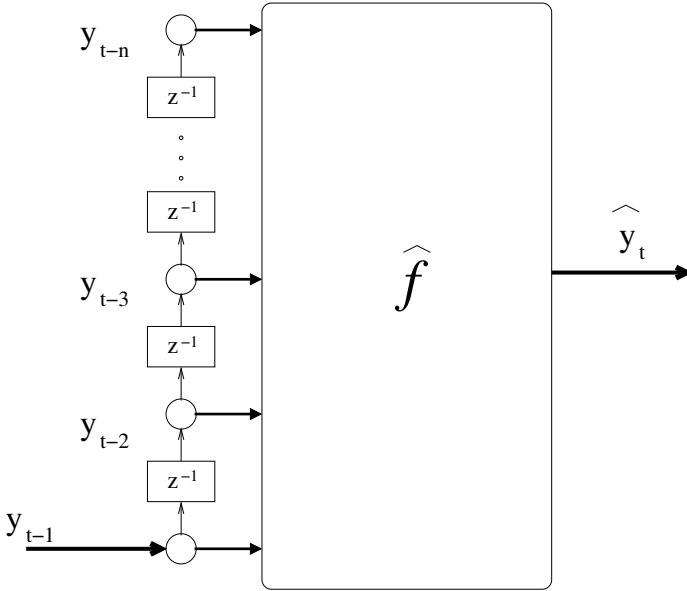


Fig. 1. One-step forecasting. The approximator \hat{f} returns the prediction of the value of the time series at time $t + 1$ as a function of the n previous values (the rectangular box containing z^{-1} represents a unit delay operator, i.e., $y_{t-1} = z^{-1}y_t$).

and the $[(N - n - 1) \times 1]$ output vector

$$Y = \begin{bmatrix} y_N \\ y_{N-1} \\ \vdots \\ y_{n+1} \end{bmatrix} \quad (7)$$

For the sake of simplicity, we assume here a $d = 0$ lag time. Henceforth, in this chapter we will refer to the i^{th} row of X , which is essentially a temporal pattern of the series, as to the (reconstructed) *state* of the series at time $t - i + 1$.

3.2 Instantiation with Local Learning

Forecasting one-step-ahead consists then in predicting the value of the output when a subset of past observed values (also denoted as query) is given. Machine learning provides a theoretical framework to estimate from observed data a suitable model of the time dependency f . Because of the impossibility of reviewing here the entire state-of-the-art of machine learning in time series forecasting, we will more specifically consider local learning techniques [12,31,29] in the following section. This choice is motivated by the following reasons:

- Reduced number of assumptions: local learning assumes no a priori knowledge on the process underlying the data. For example, it makes no

assumption on the existence of a global function describing the data and no assumptions on the properties of the noise. The only available information is represented by a finite set of input/output observations. This feature is particularly relevant in real datasets where problems of missing features, non-stationarity and measurement errors make appealing a data-driven and assumption-free approach.

- On-line learning capability: The local learning method can easily deal with on-line learning tasks where the number of training samples increases with time. In this case, local learning simply adds new points to the dataset and does not need time-consuming re-training when new data become available.
- Modelling non-stationarity: The local learning method can deal with time-varying configurations where the stochastic process underlying the data is non-stationary. In this case, it is sufficient to interpret the notion of neighbourhood not in a spatial way but both in a spatial and temporal sense. For each query point, the neighbours are no more the samples that have similar inputs but the ones that both have similar inputs and have been collected recently in time. Therefore, the time variable becomes a further precious feature to consider for accurate prediction.

We describe in the following two instances of local learning techniques, namely Nearest Neighbor [36,29] and Lazy Learning [12,5].

Nearest Neighbor. The Nearest Neighbor method is the most trivial example of local approximation applied to the problem of time series forecasting. This method consists in looking through the data set for the nearest neighbor of the current state and predicting that the current state will evolve in the same manner as the neighbor did.

Figure 2 represents an example of nearest-neighbor one-step forecasting. Suppose we have available a time series y_t up to time $\bar{t}-1$ and we want to predict the next value of the series. Once selected a certain dimension n , for example $n = 6$, the nearest neighbor approach searches for the pattern in the past which is the most similar, in a given metric, to the pattern $\{y_{\bar{t}-6}, y_{\bar{t}-5}, \dots, y_{\bar{t}-1}\}$ (the dashed line). If the nearest pattern is, for instance, $\{y_{\bar{t}-16}, y_{\bar{t}-15}, \dots, y_{\bar{t}-11}\}$, then the forecasts $\hat{y}_{\bar{t}}$ returned by the NN method is the value $y_{\bar{t}-10}$ (black dot).

This approach was first proposed by Lorenz [36] to examine weather maps. Imagine that we want to predict tomorrow's weather in Bruxelles and that we choose a dimension $n = 1$. The nearest neighbor approach suggests (i) to search the historical database of the meteorological conditions in Bruxelles, (ii) to find the weather pattern most similar to that of today (for example the weather pattern on March 5th, 1999, by chance a rainy day!) and (iii) to predict that tomorrow's weather will be the same as March 6th, 1999 (just by chance another rainy day!).

Natural extensions of the Nearest Neighbor approach consider more neighbors [31] or higher order approximations. Piecewise linear approximation in time series analysis was introduced by Tong and Lim [56]. Priestley [46] suggested the importance of higher order approximations. Farmer and Sidorowich [23,24]

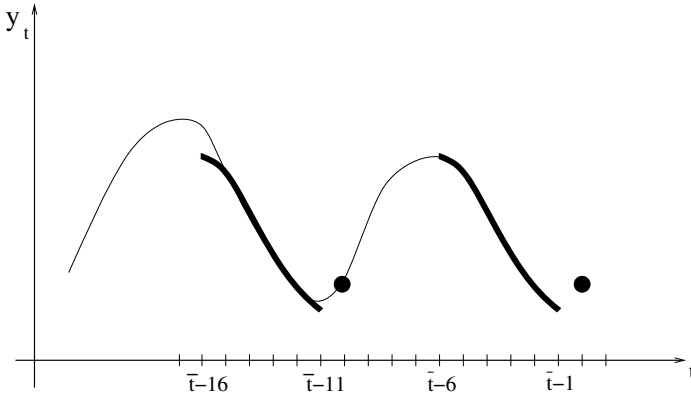


Fig. 2. Nearest-neighbor one-step-ahead forecasts. We want to predict at time $\bar{t} - 1$ the next value of the series y of order $n = 6$. The pattern $y_{\bar{t}-16}, y_{\bar{t}-15}, \dots, y_{\bar{t}-11}$ is the most similar to the pattern $\{y_{\bar{t}-6}, y_{\bar{t}-5}, \dots, y_{\bar{t}-1}\}$. Then, the prediction $\hat{y}_{\bar{t}} = y_{\bar{t}-10}$ is returned.

studied local approximation in time series and demonstrated its effectiveness on several experiments and numerical time series analysis. In particular they applied local learning techniques to predict the behavior of chaotic time series, sequences which, although deterministic, are characterized by second-order persistent statistics with random properties.

Lazy Learning. The Lazy Learning (LL) is a lazy and local learning machine [12, 11] which automatically adapts the size of the neighborhood on the basis of a cross-validation criterion. The major appeal of Lazy Learning is its divide-and-conquer nature: Lazy Learning reduces a complex and nonlinear modeling problem into a sequence of easily manageable local linear problems, one for each query. This allows to exploit, on a local basis, the whole range of linear identification and validation techniques which are fast, reliable, and come with a wealth of theoretical analyses, justifications, and guarantees. The Lazy Learning procedure essentially consists of the following steps once the matrix X in (6) and Y in (7) and a query point \mathbf{x}_q are given:

1. Sort increasingly the set of vectors in X with respect to the distance (e.g. Euclidean) to \mathbf{x}_q .
2. Determine the optimal number of neighbors.
3. Calculate, given the number of neighbors, the prediction for the query point by using a local model (e.g. constant or linear).

Let us consider a time series $\{y_1, \dots, y_t\}$ composed of t observations for which we intend to predict the next one.

The forecasting problem boils down to estimating the output \hat{y}_{t+1} when the latest window of observations is represented by the vector $\mathbf{x}_q = \{y_t, \dots, y_{t-n+1}\}$. Algorithm 1 illustrates how constant local learning techniques return the output

associated to a query point \mathbf{x}_q , for a given number of neighbors k . The notation $[j]$ is used to designate the index of the j th closest neighbor of \mathbf{x}_q . Note that also the local linear version of the algorithm is commonly used, as discussed in [11].

Algorithm 1. LL

Input : $D = \{(\mathbf{x}_i, y_i) \in (\mathbb{R}^n \times \mathbb{R})\}$, dataset.

Input : $\mathbf{x}_q \in \mathbb{R}^d$, query point.

Input : $k =$ the number of neighbors.

Output: \hat{y}_{t+1} , the estimation of the output of the query point \mathbf{x}_q
(obtained with k neighbors).

Sort increasingly the set of vectors $\{\mathbf{x}_i\}$ with respect to the distance to \mathbf{x}_q .

$$\hat{y}_{t+1} = \frac{1}{k} \sum_{j=1}^k y_{[j]}.$$

return \hat{y}_{t+1} .

This algorithm requires the choice of a set of model parameters (e.g. the number k of neighbors, the kernel function, the distance metric) [5]. We will discuss here an automatic method based on a Leave-One-Out (LOO) criterion to determine the number of neighbor [11][12]. The main idea is to assess the quality of each local model by using a LOO measure and to select the best neighborhood size according to such measure.

A computationally efficient way to perform LOO cross-validation and to assess the performance in generalization of local linear models is the PRESS statistic, proposed in 1974 by Allen [2]. By assessing the performance of each local model, alternative configurations can be tested and compared in order to select the best one in terms of expected prediction. The idea consists in associating an LOO error $e_{LOO}(k)$ to the estimation

$$\hat{y}_q^{(k)} = \frac{1}{k} \sum_{j=1}^k y_{[j]}, \quad (8)$$

associated to the query point \mathbf{x}_q and returned by k neighbors. In case of a constant model, the LOO term can be derived as follows [12]:

$$e_{LOO}(k) = \frac{1}{k} \sum_{j=1}^k (e_j(k))^2, \quad (9)$$

where

$$e_j(k) = y_{[j]} - \frac{\sum_{i=1(i \neq j)}^k y_{[i]}}{k-1} = k \frac{y_{[j]} - \hat{y}_k}{k-1}. \quad (10)$$

The best number of neighbors is then defined as the number

$$k^* = \arg \min_{k \in \{2, \dots, K\}} e_{LOO}(k), \quad (11)$$

which minimizes the LOO error.

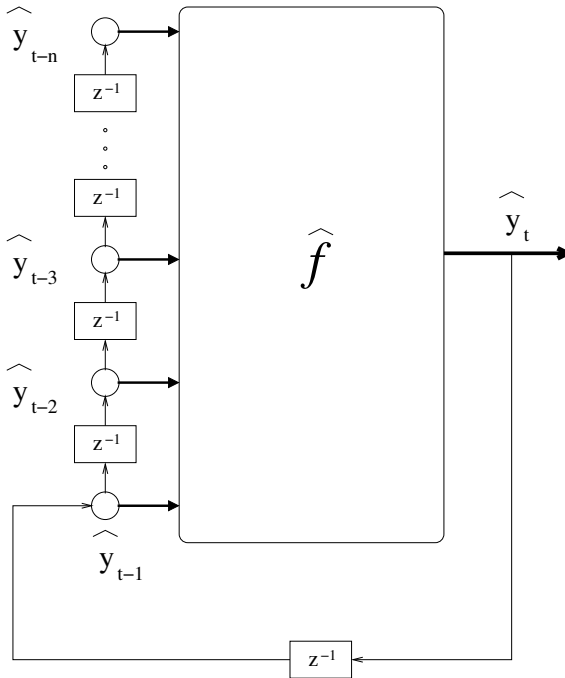


Fig. 3. Iterated prediction. The approximator \hat{f} returns the prediction of the value of the time series at time $t + 1$ by iterating the predictions obtained in the previous steps (the rectangular box containing z^{-1} represents a unit delay operator, i.e., $\hat{y}^{t-1} = z^{-1}\hat{y}^t$).

Lazy learning was applied with success to several regression and one-step forecasting tasks [14]. More details on the LL technique and its applications can be found in [11][12].

4 Strategies for Multi-step Time Series Forecasting

The previous section showed that one-step forecasting can be cast in a conventional supervised learning framework by having recourse to conventional learning techniques such as Local Learning. In this section, we extend the framework to show how learning techniques can be used to tackle the multi-step forecasting problem. Three strategies can be considered, namely recursive, direct and multiple output strategies.

A multi-step time series forecasting task consists of predicting the next H values $[y_{N+1}, \dots, y_{N+H}]$ of a historical time series $[y_1, \dots, y_N]$ composed of N observations, where $H > 1$ denotes the forecasting horizon.

This section will give a presentation of the three existing strategies to adopt machine learning in multi-step forecasting. We will use a common notation where f and F denote the functional dependency between past and future observations,

n refers to the embedding dimension [17] of the time series, that is the number of past values used to predict future values and w represents the term that includes modeling error, disturbances and/or noise.

4.1 Recursive Strategy

The *Recursive* strategy [58,49,18] trains first a one-step model f

$$y_{t+1} = f(y_t, \dots, y_{t-n+1}) + w_{t+1}, \quad (12)$$

with $t \in \{n, \dots, N - 1\}$ and then uses it recursively for returning a multi-step prediction (Figure 3). A well-known drawback of the recursive method is its sensitivity to the estimation error, since estimated values, instead of actual ones, are more and more used when we get further in the future.

In spite of these limitations, the Recursive strategy has been successfully used to forecast many real-world time series by using different machine learning models, like recurrent neural networks [47] and nearest-neighbors [38,15].

4.2 Direct Strategy

The *Direct* strategy [58,49,18] learns independently H models f_h

$$y_{t+h} = f_h(y_t, \dots, y_{t-n+1}) + w_{t+h}, \quad (13)$$

with $t \in \{n, \dots, N - H\}$ and $h \in \{1, \dots, H\}$ and returns a multi-step forecast by concatenating the H predictions.

Since the Direct strategy does not use any approximated values to compute the forecasts (Equation 13), it is not prone to any accumulation of errors. Notwithstanding, it has some weaknesses. First, since the H models are learned independently no statistical dependencies between the predictions \hat{y}_{N+h} [13,16,32] is considered. Second direct methods often require higher functional complexity [54] than iterated ones in order to model the stochastic dependency between two series values at two distant instants [27]. Last but not least, this strategy demands a large computational time since the number of models to learn is equal to the size of the horizon.

Different machine learning models have been used to implement the Direct strategy for multi-step forecasting tasks, for instance neural networks [32], nearest neighbors [49] and decision trees [57].

4.3 DirRec Strategy

The *DirRec* strategy [50] combines the architectures and the principles underlying the Direct and the Recursive strategies. DirRec computes the forecasts with different models for every horizon (like the Direct strategy) and, at each time step, it enlarges the set of inputs by adding variables corresponding to the forecasts of the previous step (like the Recursive strategy). However, note that unlike the two previous strategies, the embedding size n is not the same for all

the horizons. In other terms, the DirRec strategy learns H models f_h from the time series $[y_1, \dots, y_N]$ where

$$y_{t+h} = f_h(y_{t+h-1}, \dots, y_{t-n+1}) + w_{t+h}, \quad (14)$$

with $t \in \{n, \dots, N - H\}$ and $h \in \{1, \dots, H\}$.

4.4 Multiple Output Strategies

In spite of their diversity, iterated and direct techniques for multiple-step forecasting share a common feature: they model from data a multi-input single-output mapping whose output is the variable y_{t+1} in the iterated case and the variable y_{t+k} in the direct case, respectively. When a very long term prediction is at stake and a stochastic setting is assumed, the modeling of a single-output mapping neglects the existence of stochastic dependencies between future values, (e.g. between y_{t+k} and y_{t+k+1}) and consequently biases the prediction accuracy. A possible way to remedy to this shortcoming is to move from the modeling of single-output mappings to the modeling of multi-output dependencies. This requires the adoption of multi-output techniques where the predicted value is no more a scalar quantity but a vector of future values of the time series.

The MIMO Strategy. The *Multi-Input Multi-Output* (MIMO) strategy [13,16] (also known as Joint strategy [32]) avoids the simplistic assumption of conditional independence between future values made by the Direct strategy [13,16] by learning a single multiple-output model

$$[y_{t+H}, \dots, y_{t+1}] = F(y_t, \dots, y_{t-n+1}) + \mathbf{w}, \quad (15)$$

where $t \in \{n, \dots, N - H\}$, $F : \mathbb{R}^d \rightarrow \mathbb{R}^H$ is a vector-valued function [39], and $\mathbf{w} \in \mathbb{R}^H$ is a noise vector with a covariance that is not necessarily diagonal [37].

The forecasts are returned in one step by a multiple-output model \hat{F} where

$$[\hat{y}_{t+H}, \dots, \hat{y}_{t+1}] = \hat{F}(y_N, \dots, y_{N-n+1}). \quad (16)$$

The rationale of the MIMO strategy is to model, between the predicted values, the stochastic dependency characterizing the time series. This strategy avoids the conditional independence assumption made by the Direct strategy as well as the accumulation of errors which plagues the Recursive strategy. So far, this strategy has been successfully applied to several real-world multi-step time series forecasting tasks [13,16,10,9].

However, the wish to preserve the stochastic dependencies constrains all the horizons to be forecasted with the same model structure. Since this constraint could reduce the flexibility of the forecasting approach [10], a variant of the MIMO strategy is discussed in the following section.

The DIRMOMO Strategy. The DIRMOMO strategy [10,9] aims to preserve the most appealing aspects of DIRECT and MIMO strategies by partitioning the horizon H in several blocks, and using MIMO to forecast the values inside each

block. This means that the H -step forecast requires m multiple-output forecasting tasks ($m = \frac{H}{s}$), each having an output of size s ($s \in \{1, \dots, H\}$).

Note that for $s = 1$, the DIRMO coincides with the conventional Direct strategy, while for $s = H$ it corresponds to the MIMO strategy. The tuning of the parameter s allows us to improve the flexibility of the MIMO strategy by calibrating the dimensionality of the outputs (no dependency in the case $s = 1$ and maximal dependency for $s = H$). This provides a beneficial trade off between the preserving a larger degree of the stochastic dependency between future values and having a greater flexibility of the predictor.

5 Local Learning for Multi-step Forecasting

Local learning appears to be an effective algorithm not only for one-step but also for multi-step forecasting. This section discusses some works which used local learning techniques to deal specifically with the long term forecasting problem.

In [38,15] the authors proposed a modification of the local learning technique to take into account the temporal behavior of the multi-step forecasting problem and consequently improve the results of the recursive strategies. In particular [15] modified the PRESS criterion ([10]) by introducing an iterated version of the leave-one-out statistic. They showed that the iterated PRESS outperforms a non-iterated criterion by assessing the generalization performance of a local one-step predictor on a horizon longer than a single step, yet preserving nice properties of computational efficiency. It is worth noting that the two techniques proposed by [38] and [15] ranked respectively first and second in the 1998 Leuven time series prediction.

A recent improvement of the recursive strategy based again on local learning is RECNOISY [6], which perturbs the initial dataset at each step of the forecasting process to handle more properly the approximated values in the prediction process. The rationale of the RECNOISY method is that the training examples used by the recursive strategy, though observed, are not necessarily representative of the forecasting tasks which will be required later all along the forecasting process. To remedy to this problem, this strategy exploits the particular nature of the forecasting tasks induced by the recursive strategy and incorporates it in the local learning phase in order to improve the results.

Two improvements of Lazy Learning to deal with long-term prediction of time series are presented in [51]. The first method is based on an iterative pruning of the inputs; the second one performs a brute force search in the possible set of inputs using a k-NN approximator.

The use of local learning for multi-input multi-output prediction was proposed in [13] where a multi-output extension of the algorithm [1] is discussed as well as an averaging strategy of several long term predictors to improve the resulting accuracy.

The use of the local learning approximator to implement a DIRMO strategy is presented in [8,9]. The DIRMO strategy based on local learning has been successfully applied to two forecasting competitions: ESTSP'07 [10] and NN3 [9].

A detailed review and comparison of strategies for multi-step time series forecasting based on the local learning algorithm is presented in [7].

6 Conclusion

Predicting the future is one of the most relevant and challenging tasks in applied sciences. Building effective predictors from historical data demands computational and statistical methods for inferring dependencies between past and short-term future values of observed values as well as appropriate strategies to deal with longer horizons. This chapter discussed the role of machine learning in adapting supervised learning techniques to deal with forecasting problems. In particular we stressed the role played by local learning approximators in dealing with important issues in forecasting, like nonlinearity, nonstationarity and error accumulation. Future research should be concerned with the extension of these techniques to some recent directions in business intelligence, like the parallel mining of huge amount of data (big data) [41] and the application to spatio-temporal tasks [30].

Acknowledgments. Gianluca Bontempi acknowledges the support of the ARC project "Discovery of the molecular pathways regulating pancreatic beta cell dysfunction and apoptosis in diabetes using functional genomics and bioinformatics" funded by the Communauté Française de Belgique.

References

1. Ahmed, N.K., Atiya, A.F., El Gayar, N., El-Shishiny, H.: An empirical comparison of machine learning models for time series forecasting. *Econometric Reviews* 29(5-6) (2010)
2. Allen, D.M.: The relationship between variable selection and data augmentation and a method for prediction. *Technometrics* 16(1), 125–127 (1974)
3. Alpaydin, E.: *Introduction to Machine Learning*, 2nd edn. Adaptive Computation and Machine Learning. The MIT Press (February 2010)
4. Anderson, T.W.: *The statistical analysis of time series*. J. Wiley and Sons (1971)
5. Atkeson, C.G., Moore, A.W., Schaal, S.: Locally weighted learning. *AIR* 11(1-5), 11–73 (1997)
6. Ben Taieb, S., Bontempi, G.: Recursive multi-step time series forecasting by perturbing data. In: *Proceedings of IEEE-ICDM 2011*(2011)
7. Ben Taieb, S., Bontempi, G., Atiya, A., Sorjamaa, A.: A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. *ArXiv e-prints* (August 2011)
8. Ben Taieb, S., Bontempi, G., Sorjamaa, A., Lendasse, A.: Long-term prediction of time series by combining direct and mimo strategies. In: *Proceedings of the 2009 IEEE International Joint Conference on Neural Networks*, Atlanta, U.S.A., pp. 3054–3061 (June 2009)
9. Ben Taieb, S., Sorjamaa, A., Bontempi, G.: Multiple-output modelling for multi-step-ahead forecasting. *Neurocomputing* 73, 1950–1957 (2010)

10. Ben Taieb, S., Bontempi, G., Sorjamaa, A., Lendasse, A.: Long-term prediction of time series by combining direct and mimo strategies. In: International Joint Conference on Neural Networks (2009)
11. Birattari, M., Bontempi, G., Bersini, H.: Lazy learning meets the recursive least-squares algorithm. In: Kearns, M.S., Solla, S.A., Cohn, D.A. (eds.) NIPS 11, pp. 375–381. MIT Press, Cambridge (1999)
12. Bontempi, G.: Local Learning Techniques for Modeling, Prediction and Control. PhD thesis, IRIDIA- Université Libre de Bruxelles (1999)
13. Bontempi, G.: Long term time series prediction with multi-input multi-output local learning. In: Proceedings of the 2nd European Symposium on Time Series Prediction (TSP), ESTSP 2008, Helsinki, Finland, pp. 145–154 (February 2008)
14. Bontempi, G., Birattari, M., Bersini, H.: Lazy learners at work: the lazy learning toolbox. In: Proceeding of the 7th European Congress on Intelligent Techniques and Soft Computing, EUFIT 1999 (1999)
15. Bontempi, G., Birattari, M., Bersini, H.: Local learning for iterated time-series prediction. In: Bratko, I., Dzeroski, S. (eds.) Machine Learning: Proceedings of the Sixteenth International Conference, pp. 32–38. Morgan Kaufmann Publishers, San Francisco (1999)
16. Bontempi, G., Ben Taieb, S.: Conditionally dependent strategies for multiple-step-ahead prediction in local learning. *International Journal of Forecasting* (2011) (in press, corrected proof)
17. Casdagli, M., Eubank, S., Farmer, J.D., Gibson, J.: State space reconstruction in the presence of noise. *PHYSICA D* 51, 52–98 (1991)
18. Cheng, H., Tan, P.-N., Gao, J., Scripps, J.: Multistep-Ahead Time Series Prediction. In: Ng, W.-K., Kitsuregawa, M., Li, J., Chang, K. (eds.) PAKDD 2006. LNCS (LNAI), vol. 3918, pp. 765–774. Springer, Heidelberg (2006)
19. Crone, S.F.: NN3 Forecasting Competition, <http://www.neural-forecasting-competition.com/NN3/index.html> (last update May 26, 2009) (visited on July 05, 2010)
20. Crone, S.F.: NN5 Forecasting Competition, <http://www.neural-forecasting-competition.com/NN5/index.html> (last update May 27, 2009) (visited on July 05, 2010)
21. Crone, S.F.: Mining the past to determine the future: Comments. *International Journal of Forecasting* 5(3), 456–460 (2009); Special Section: Time Series Monitoring
22. Engle, R.F.: Autoregressive conditional heteroscedasticity with estimates of the variance of united kingdom inflation. *Econometrica* 50(4), 987–1007 (1982)
23. Farmer, J.D., Sidorowich, J.J.: Predicting chaotic time series. *Physical Review Letters* 8(59), 845–848 (1987)
24. Farmer, J.D., Sidorowich, J.J.: Exploiting chaos to predict the future and reduce noise. Technical report, Los Alamos National Laboratory (1988)
25. De Gooijer, J.G., Hyndman, R.J.: 25 years of time series forecasting. *International Journal of Forecasting* 22(3), 443–473 (2006)
26. De Gooijer, J.G., Kumar, K.: Some recent developments in non-linear time series modelling, testing, and forecasting. *International Journal of Forecasting* 8(2), 135–156 (1992)
27. Guo, M., Bai, Z., An, H.Z.: Multi-step prediction for nonlinear autoregressive models based on empirical distributions. In: *Statistica Sinica*, pp. 559–570 (1999)
28. Hand, D.: Mining the past to determine the future: Problems and possibilities. *International Journal of Forecasting* (October 2008)

29. Hastie, T., Tibshirani, R., Friedman, J.: The elements of statistical learning: data mining, inference and prediction, 2nd edn. Springer (2009)
30. Hsu, W., Lee, M.L., Wang, J.: Temporal and spatio-temporal data mining. IGI Pub. (2008)
31. Ikeguchi, T., Aihara, K.: Prediction of chaotic time series with noise. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences E78-A(10) (1995)
32. Kline, D.M.: Methods for multi-step time series forecasting with neural networks. In: Peter Zhang, G. (ed.) Neural Networks in Business Forecasting, pp. 226–250. Information Science Publishing (2004)
33. Lapedes, A., Farber, R.: Nonlinear signal processing using neural networks: prediction and system modelling. Technical Report LA-UR-87-2662, Los Alamos National Laboratory, Los Alamos, NM (1987)
34. Lendasse, A. (ed.): ESTSP 2007: Proceedings (2007)
35. Lendasse, A. (ed.): ESTSP 2008: Proceedings. Multiprint Oy/Otamedia (2008) ISBN: 978-951-22-9544-9
36. Lorenz, E.N.: Atmospheric predictability as revealed by naturally occurring analogues. *Journal of the Atmospheric Sciences* 26, 636–646 (1969)
37. Matías, J.M.: Multi-output Nonparametric Regression. In: Bento, C., Cardoso, A., Dias, G. (eds.) EPIA 2005. LNCS (LNAI), vol. 3808, pp. 288–292. Springer, Heidelberg (2005)
38. McNames, J.: A nearest trajectory strategy for time series prediction. In: Proceedings of the International Workshop on Advanced Black-Box Techniques for Nonlinear Modeling, pp. 112–128. K.U. Leuven, Belgium (1998)
39. Micchelli, C.A., Pontil, M.A.: On learning vector-valued functions. *Neural Comput.* 17(1), 177–204 (2005)
40. Mitchell, T.M.: Machine Learning. McGraw-Hill, New York (1997)
41. Owen, S.: Mahout in action. Manning (2012)
42. Packard, N.H., Crutchfield, J.P., Farmer, J.D., Shaw, R.S.: Geometry from a time series. *Physical Review Letters* 45(9), 712–716 (1980)
43. Palit, A.K., Popovic, D.: Computational Intelligence in Time Series Forecasting: Theory and Engineering Applications. Advances in Industrial Control. Springer-Verlag New York, Inc., Secaucus (2005)
44. Poskitt, D.S., Tremayne, A.R.: The selection and use of linear and bilinear time series models. *International Journal of Forecasting* 2(1), 101–114 (1986)
45. Price, S.: Mining the past to determine the future: Comments. *International Journal of Forecasting* 25(3), 452–455 (2009)
46. Priestley, M.B.: Non-linear and Non-stationary time series analysis. Academic Press (1988)
47. Saad, E., Prokhorov, D., Wunsch, D.: Comparative study of stock trend prediction using time delay, recurrent and probabilistic neural networks. *IEEE Transactions on Neural Networks* 9(6), 1456–1470 (1998)
48. Schuster, H.G.: Deterministic Chaos: An Introduction. Weinheim Physik (1988)
49. Sorjamaa, A., Hao, J., Reyhani, N., Ji, Y., Lendasse, A.: Methodology for long-term prediction of time series. *Neurocomputing* 70(16-18), 2861–2869 (2007)
50. Sorjamaa, A., Lendasse, A.: Time series prediction using dirrec strategy. In: Verleysen, M. (ed.) European Symposium on Artificial Neural Networks, ESANN 2006, Bruges, Belgium, April 26-28, pp. 143–148 (2006)
51. Sorjamaa, A., Lendasse, A., Verleysen, M.: Pruned lazy learning models for time series prediction. In: European Symposium on Artificial Neural Networks, ESANN 2005, pp. 509–514 (2005)

52. Takens, F.: Detecting strange attractors in fluid turbulence. In: *Dynamical Systems and Turbulence*. Springer, Berlin (1981)
53. Tiao, G.C., Tsay, R.S.: Some advances in non-linear and adaptive modelling in time-series. *Journal of Forecasting* 13(2), 109–131 (1994)
54. Tong, H.: *Threshold models in Nonlinear Time Series Analysis*. Springer, Berlin (1983)
55. Tong, H.: *Non-linear Time Series: A Dynamical System Approach*. Oxford University Press (1990)
56. Tong, H., Lim, K.S.: Threshold autoregression, limit cycles and cyclical data. *JRSS_B* 42, 245–292 (1980)
57. Tran, T.V., Yang, B.-S., Tan, A.C.C.: Multi-step ahead direct prediction for the machine condition prognosis using regression trees and neuro-fuzzy systems. *Expert Syst. Appl.* 36(5), 9378–9387 (2009)
58. Weigend, A.S., Gershenfeld, N.A.: *Time Series Prediction: forecasting the future and understanding the past*. Addison Wesley, Harlow (1994)
59. Werbos, P.J.: *Beyond Regression: New Tools for Prediction and Analysis in the Behavioral Sciences*. PhD thesis, Harvard University, Cambridge, MA (1974)
60. Werbos, P.J.: Generalization of backpropagation with application to a recurrent gas market model. *Neural Networks* 1(4), 339–356 (1988)
61. Zhang, G., Eddy Patuwo, B., Hu, M.Y.: Forecasting with artificial neural networks: The state of the art. *International Journal of Forecasting* 14(1), 35–62 (1998)

Knowledge Discovery from Constrained Relational Data: A Tutorial on Markov Logic Networks

Marcus Spies

Knowledge Management
LMU University of Munich, Germany
marcus.spies@ieee.org

Abstract. This tutorial paper gives an overview of Markov logic networks (MLNs) in theory and in practice. The basic concepts of MLNs are introduced in a semi-formal way and examined for their significance in the broader context of statistical relational learning approaches in general and Bayesian logic networks in particular. A sandbox example is discussed in order to explain in detail the meanings of input theories with weighted clauses for a MLN. Then, the setup needed for real-world applications using a recent open source prototype is introduced. Processing steps of inferencing and learning are explained in detail together with the best scaling algorithms known today. An overview on existing and upcoming application areas concludes the paper.

1 Introduction

We briefly recapitulate the history of both Bayesian and Markov logic networks.

Bayesian networks became popular in the eighties of the 20th century as a powerful mechanism for integrating probabilistic models with plausible inference [37]. In 1988, the scalable algorithm by Lauritzen and Spiegelhalter for flexible inference processing in Bayesian networks, integrating concepts from Markov random fields, became available [28]. In many subsequent medical and industrial applications, Bayesian networks were used with expert provided probability tables and expert provided network structure. An early successful application of Bayesian networks was causal analysis of production defects detection at microchip manufacturer Intel. In the following years, the methodology of Bayesian networks was successively extended to cover, first, learning of conditional probability tables from data, and, later, extensions for learning structure of Bayesian networks under various optimization heuristics, see [6] and [13].

In the past decade, the inherent limitation of Bayesian networks to single entities or observation units was increasingly perceived as a barrier to further applications in upcoming disciplines such as genome research and Web mining, text mining etc. In two seminal papers, [14] and [27], the formalism of Bayesian networks was extended to 1st order logic, implying the applicability of Bayesian networks to models with several related entities, as they are common in, e.g.,

entity relationship models. These approaches became part of a greater stream of innovative research on *Statistical Relational Learning* that included new approaches like conditional random fields (CRF). The reference book publication in the SRL area was [11]. For an excellent recent overview of approaches to probabilistic inference in knowledge-based information systems, see [20].

Bayesian logic networks (BLNs) draw from both these approaches. The theoretical rationale can be found in the chapter [21] which also provides a modelling approach and a tool that integrates a Prolog engine for inferencing. [32] contributes a domain specific modelling language that supports the definition of multi-entity models with stochastic dependencies (see below section 4). Finally, [18] provide a graphical modelling and processing tool that integrates Markov logic network functionality, as well.

Markov random fields originate from statistical mechanics. The application in the area of probabilistic inference in expert systems was initiated by research on Boltzmann machines in the 80s and 90s of the last century [3]. Boltzmann machines were successfully applied in several combinatorial optimization [2] and cognitive modelling tasks [49]. A learning algorithm that would enable to estimate connectivity between processing units for Boltzmann machine became available [4] and was extended to incomplete patterns in [50]. However, scalability remained an issue for implementing both Boltzmann machine learning and inference. Furthermore, the inherent limitation imposed by binary-state processing units was equivalent to the limitation of Bayesian networks to single entities (which could be equivalently described by a single binary vector for each observation unit in the case of dichotomous variables).

In an analogous way to Bayesian logic networks, Markov logic networks (MLNs) were designed to enable the use of first order logic input theories defining and / or constraining a multi-entity probabilistic graphical model. MLNs were introduced in [40] and [10], a reference implementation was made available early on by the ALCHEMY project [25]. Besides the extended expressivity allowed by first order theories, an important key contribution in MLNs was related to scalability addressed across all phases of processing. Recently, the TUFFY project [36] has contributed further improvements in scalability and extensibility.

Common to both approaches is the idea of basing a model for a given domain on a general input theory definition that includes statements expressing uncertainty as well as logical theorems and constraints. If a specific data set becomes available, the model constants become interpreted and the statements of the theory are compiled into a ground network, which is a standard Bayesian network or Markov random field depending on the MLN or BLN approach being taken. Also common to both approaches is the capability of parameter and structure learning. The key difference between the approaches lies in the nature of uncertainty being expressed in the input theories and the resulting grounded models. This will be explained and discussed in detail below.

In the present tutorial, we will focus on MLN. Some explanations will be given in order to distinguish them more clearly from the BLN approach in section 3.

2 Markov Logic Network (MLN) Concepts

In this section we first explain the basic definition and processing steps in a Markov Logic Network (MLN) in subsection 2.1. The key input to running an MLN is a set of formulae in predicate logic attributed with weights which express a degree of uncertainty. This input gives rise to a stochastic model of any specific application domain on which the MLN processes various inference procedures. The model and some procedures are illustrated by a few elementary examples in subsection 2.2. Later, in section 5.3, we will address options for learning of MLN inputs from data.

2.1 Basic Definition and Processing Steps of a Markov Logic Network (MLN)

Input to a Markov Logic Network (MLN) is a first-order logic (FOL) theory Θ expressed as a finite set of FOL formulae $F_i, i \in \{1, \dots, m\}$. For definitions and illustrative examples of FOL theories, the reader is invited to consult a textbook like [31,44,45]. It is generally assumed that these formulae are preprocessed according to the usual rules for generating conjunctive normal forms (CNF, see [31,45]). As a result, all F_i are quantifier-free and contain only disjunctions of literals. A literal is an atomic formula consisting of a predicate with a suitable number of arguments. A literal may appear in positive or negated form in a formula. An argument to a predicate in a literal is also called a term. The terms in literals consist of individual variables or individual constants which may appear nested in function-symbols. Function symbols can be genuine elements of a theory, but they may also be generated through the elimination of existential quantifiers in the process of constructing a CNF [31,45].

Restrictions on the admissible structures of input formulae need to be chosen depending on the specific application and computational paradigm the input theory is related to. Generic FOL theories have well known decidability and tractability issues. Therefore, in most applications, restricted subsets of FOL are enforced. The best known such subsets are related to variants of description logic DL [5]. DL is also the core of current web ontology languages like OWL2 [33]. Therefore, a critical issue in defining an FOL theory for an MLN is the appropriate choice of an input theory definition language. We will review some approaches to definition languages for MLN in section 4.

Each formula $F_i, i \in \{1, \dots, m\}$ is assigned either a weight $w(F_i) \in \mathbb{R}$, or it is assumed to hold with certainty.

It is assumed that there is a monotonically increasing relationship between weights and probabilities of formula instances, such that a formula with heigher weight should have a higher probability of verifying instances in all possible interpretations of the input theory. It should be noted, however, that there is no generic complementarity of a weight for a formula as compared to the weight for its negation. This is one key distinction against the Bayesian logic network approach, see below.

In most implementations, a default weight of 0 is assumed to represent indifference regarding the possible probability of instances of the formula. In many cases, this translates to an a priori probability of .5 for the set of the formula instances.

Let us now assume a specific interpretation of the theory defined by the F_i , $i \in \{1, \dots, m\}$ to be given. For MLNs and BLNs, such an interpretation \mathcal{I} comprises a *finite domain*, i.e., a *finite set of individuals* \mathcal{D} . For convenience, constants in the theory are assumed to be mapped to different such individuals as if they were uniquely named by the constants from the theory (this is the *unique name* assumption in [40]). Moreover, function symbols are assumed to have known interpretations for these named individuals, which implies that functions are pointing to named individuals again (this is the *known functions* assumption in [40]). Formally, however, these named individuals appearing as function values need not be constants in the underlying theory Θ . An even more restrictive assumption is to postulate that \mathcal{D} is composed *only* of these named individuals and their known functionally dependent counterparts (this is the *domain closure* assumption in [40]). This assumption allows to avoid complexity issues resulting from (possibly iterative) applications of functions to unknown individuals.

Taken together, these assumptions guarantee that the *Herbrand universe* of a base theory Θ is *finite and consists of symbols denoting named individuals and known functional dependent individuals only*. It should be noted here, that in the process of learning a MLN from data the base universe may of course change and that the parameters of the probability representation of the uncertainty model (see below) are independent of the specific set of individuals being chosen in one interpretation domain \mathcal{D} .

Possible relaxations of these assumptions will be discussed in subsection 4.4.

Given a Herbrand universe constructed from \mathcal{I} under these assumptions, formula F_i may be true for some individuals and false for others. We call the set of ground formulae derived from our theory Θ by applying all possible substitutions of named domain individuals from \mathcal{D} for variables the *Herbrand interpretation* of Θ . Then, a *Herbrand model* of our theory Θ in \mathcal{I} is a subset of its Herbrand interpretation that is logically entailed by Θ . For brevity, we refer to elements of the Herbrand interpretation of Θ as *formula instances*. In particular, the set of ground instances of literals in the input theory is referred to as the *Herbrand base* of Θ in \mathcal{I} .

The purpose of the probability model corresponding to a MLN is to describe probabilities of true ground instances of theorems of a given input theory in a given interpretation. By the parameterization described below, these interpretation specific probabilities are generalized to interpretation independent probabilities. The probability model for MLN rests on the assumption that each element of the aforementioned *Herbrand base* may be independently set to true or to false at random as long as the input theory does not imply a dependency between the truth values of several such elements. Such a dependency corresponds to a logical connective in one or more of the input theorems, and the input weights correspond to strengths of such dependencies.

These assumptions lead to a *Markov random field* probability model for Θ in \mathcal{I} . This model can be conveniently derived starting from an undirected graph with a node for each element of the *Herbrand base* of the given theory (i.e., a node for each interpreted literal). The node represents a Boolean random variable whose outcomes are truth (1) or falsity (0) of the respective interpreted literal. Two nodes are connected if a formula being interpreted contains both literals. The undirected graph constructed from Θ in \mathcal{I} is commonly referred to as the *ground Markov network* corresponding to the input theory and interpretation in question.

In this representation, each ground instance of a formula from the input theory Θ within interpretation \mathcal{I} corresponds to a clique (complete subgraph) on the corresponding ground atoms in the ground Markov network. – Another way to define the ground Markov network is to use the notion of a *hypergraph* (hypergraphs are commonly used in database schema theory). In this approach, nodes are defined as before, but hyperedges are constructed for all non-empty subsets of nodes sharing a common ground clause. In this perspective, each ground instance of a formula as described above is represented as a hyperedge in the underlying Markov network.

Next, we introduce possible worlds. From the logical point of view, a possible world is an assignment of truth values to the elements of the *Herbrand base* of Θ within interpretation \mathcal{I} . (It should be noted here that possible worlds are defined w.r.t. to a specific interpretation of the input theory as described. This is somewhat different from the tradition in formal logic, where possible worlds in first order logic are defined as laws of existence or state descriptions [15] specified without reference to a particular interpretation and its domain.)

In the Markov ground network representation, since there is a bijection between the elements of the *Herbrand base* and the nodes in this network, a possible world \mathbf{p} is a state vector of the Boolean variables with a component for each node. According to common terminology in Markov random fields, each value of \mathbf{p} uniquely represents a *configuration* of the network. In the sequel, we will use the terminology *configuration vector* or simply configuration to denote the representation of a possible world in the ground Markov network.

In order to express the satisfaction of a formula in \mathbf{p} in a short way, it is usual to introduce a so-called *feature function* $f_{ij}(\mathbf{p})$ for each formula F_i , whose domain is exactly the set of states of the j -th clique or hyperedge corresponding to the j -th formula instance of F_i and whose range is Boolean, interpreted as integers $\{0, 1\}$. This function assumes value 1 for a state of the clique or hyperedge j if the truth values of the nodes in j make F_i true. In this case, we call the clique *active* in the possible world or configuration in question. Intuitively, the feature functions we use in MLNs are simply truth tables for formula ground instances expressed with integers $\{0, 1\}$.

A feature function together with the formula weight $w_i f_{ij}(\mathbf{p})$ is in fact a *clique potential* since it provides a real-valued value depending on the states of one clique in the network (corresponding to truth / falsehood of one ground instance of formula F_i). Taking together all clique potentials, we obtain a *nearest*

neighbour potential as the sum of clique potentials in one possible world \mathbf{p} . Such a function is often referred to as a Gibbs potential. However, please note that in many references on MRF and MLN the term *potential* is actually used for the exponentiated potential as it will appear in the possible world probability below.

Finally, a ground Markov network constructed along these lines can carry a probability representation in terms of a Markov random field (MRF). In order to define the probability law for a ground Markov network, a distribution law must be defined. This law specifies the probability of any particular possible world \mathbf{p} in which all ground atoms represented as nodes in the ground Markov logic network are assigned a truth value.

Given that we assume truth and falsehood for each ground clause to be independently realizable by some stochastic process, the sum in a Gibbs potential naturally corresponds to a product of exponentials contributing to the probability of a possible world. This gives rise to a Markov random field (MRF) probability law for our ground Markov network. The overall probability of an MRF configuration or, equivalently, a single possible world \mathbf{p} can be written as

$$\Pr(\mathbf{p}) = \frac{1}{Z} \exp\left(\sum_{F_i; i=1}^n \sum_{j \in \mathcal{I}(F_i)} w_i f_{ij}(\mathbf{p})\right)$$

where Z is the normalizing sum of all exponentiated potentials on active cliques across all possible worlds (the name Z originates from the German word *Zusatzsumme* established in the original work introducing a special case of nearest neighbour potentials by physicist Ising).

$$Z = \sum_{\mathbf{p} \in \mathcal{C}} \exp\left(\sum_{F_i; i=1}^n \sum_{j \in \mathcal{I}(F_i)} w_i f_{ij}(\mathbf{p})\right)$$

Here, \mathcal{C} denotes the set of all configurations in our ground MRF. The subscript $j \in \mathcal{I}(F_i)$ is a loose way to express iteration over cliques in the ground Markov network corresponding to formula F_i .

It should be noted that $f_{ij}(\mathbf{p})$ equals zero for all cliques or hyperedges in which, for the given configuration vector \mathbf{p} , formula F_i is false. This means that all these cliques do not contribute probability mass to $\Pr(\mathbf{p})$.

Furthermore, in order to ensure $Z \neq 0.0$, we excluded negative, and, by symmetry, also positive infinite weights from our definitions for MLN input theories. This is the usual strict positivity restriction on Gibbs potentials. As we will demonstrate in the example section below, cases of formulae known to hold or te excluded with certainty can be dealt with in the construction phase of the ground Markov network.

Together, these two formulas define a discrete probability measure \Pr on the set algebra of possible worlds in our interpreted theory, or, equivalently, configurations in the resp. ground MRF. To verify this, consider the standard Kolmogorov axioms for a probability measure on a triple $(\Omega, \mathcal{A}, \Pr)$ with event set Ω , σ -algebra \mathcal{A} , and note that each configuration corresponds to an atomic event,

unions and intersections on finite event sets are defined as usually in set theory. The empty configuration (all ground atoms false) as well as the full configuration (all ground atoms true) are included in the set of configurations. Moreover, our measure \Pr is non-negative trivially, and it is additive over disjoint sets of configurations. Finally, normalization is guaranteed by dividing all likelihoods by the *Zustandssumme* Z .

Therefore, the Markov ground network together with a probability law as just described is often referred to as ground Markov random field (MRF) in the MLN literature.

Both formulas involved in the definition of the probability measure on our ground MRF can be simplified if one takes into account that the same weight w_i is applied for each true ground instance of a formula. Counting these instances to be $n_i(\mathbf{p})$ (note the count depends on the possible world we assume), we have

$$\Pr(\mathbf{p}) = \frac{1}{Z} \exp\left(\sum_{F_i; i=1}^n n_i(\mathbf{p})w_i\right).$$

However, from the computational point of view, counting n_i for any possible world requires appropriate inference steps traversing the ground network. We will discuss the complexity involved in section [5.2](#).

Given the probability of a possible world, we are ready to compute the probability of a particular ground formula \mathbf{g}_F for any specific $F \in \Theta$. Any ground formula can be true in several possible worlds. Since these correspond to disjoint atomic events in our probabilistic event algebra, the probability of a ground formula is given formally in a straightforward way by summing up the probabilities of all possible worlds in which the ground formula \mathbf{g}_F is true. This condition means, in usual terminology of mathematical logic, that \mathbf{p} is a model of \mathbf{g}_F . This will be written as $\mathbf{p} \in \mathcal{M}(\mathbf{g}_F)$. Using this notation, we can write

$$\Pr(\mathbf{g}_F) = \sum_{\mathbf{p} \in \mathcal{M}(\mathbf{g}_F)} \Pr(\mathbf{p})$$

Computing this probability in practice requires to find satisfying possible worlds for a given ground formula. This is equivalent to the \mathcal{NP} -complete SAT satisfiability problem. Therefore, for MLNs of practically relevant sizes, specific algorithms will be needed to compute this probability.

2.2 A Simple Example

As the present paper is mainly aiming at a tutorial introduction to MLNs, we give a very simple example for illustrating the concepts and the computations as introduced. Two versions of the example will be discussed in order to convey a feeling of the kind of problems to which MLNs are suitably applied.

In this paragraph, all results reported were generated with Mathematica 8 scripts written by the author. These scripts are comparatively compact and run very fast due to the built-in propositional logic syntax and corresponding satisfiability checking algorithms in Mathematica.

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|--------------------------|
| T | T | T | T | 4.6 |
| T | F | T | T | 4.3 |
| T | T | F | T | 2.6 |
| T | F | F | T | 2.3 |
| T | T | T | F | 2.6 |
| T | F | T | F | 4.3 |
| T | T | F | F | 0.6 |
| T | F | F | F | 2.3 |
| $Z = 681.322$ | | | | $\Pr\{A(a)\} = 0.433836$ |

(a) Case I

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|--------------------------|
| T | T | T | T | 4.6 |
| F | T | T | T | 4.3 |
| T | F | T | T | 4.3 |
| F | F | T | T | 4. |
| T | T | T | F | 2.6 |
| F | T | T | F | 2.3 |
| T | F | T | F | 4.3 |
| F | F | T | F | 4. |
| $Z = 681.322$ | | | | $\Pr\{B(a)\} = 0.665203$ |

(b) Case II

Fig. 1. Marginal probability results for ground network from Fig. 2 for two ground atoms, $A(a)$ in part 1a and $B(a)$ in part 1b. In these and subsequent tables, we report sums of weights for those possible worlds in which the ground atoms being examined are true. As a result, some possible worlds appear in both tables, while others are not mentioned in either table.

Let us first consider a very common problem in inference building on the traditional *modus ponens*. We assume the following theory to be given, with uncertainty on both the antecedent and the implication expressed by weights w

$$\begin{aligned}
 &F_1 \ w_1 \ A(x) \\
 &F_2 \ w_2 \ A(x) \Rightarrow B(x)
 \end{aligned}$$

Let us assume a domain consisting of two individuals a and b , $\mathcal{D} = \{a, b\}$. As a possible scenario, imagine A to mean “eats apple” and B to mean “loses permission to stay in paradise”, and take a, b to represent the first two human beings as described in the book of Genesis.

Then, the ground Markov network resulting from these assumptions is still easy to construct by hand. We need four nodes for ground atoms $A(a), A(b), B(a), B(b)$. The connectivity between these nodes is easily seen to look as depicted in Fig. 2.

Let us examine the marginal probabilities of atoms $A(a)$ and $B(a)$ for this theory, assuming weights $w_1 = .3$ and $w_2 = 2.0$. Together with the applicable possible worlds and the raw sum of weights of satisfied ground formulas for each case, figures are reported in tables in Fig. 1. The reader should verify how these sums can be explained from contributions of the single weights of those ground formulas that are true in a given possible world. The bottom line reports the “Zustandssumme” together with the marginal probability. The notation $\Pr\{A(a)\}$ means – the probability of the “event” $\{A(a)\}$ which is true in all possible worlds listed in the table depicted in Fig. 1a. Similar remarks apply to subsequent tables.

A variation of this problem can be constructed by allowing the influence of the conditional to include not only the same but also other individuals. In our

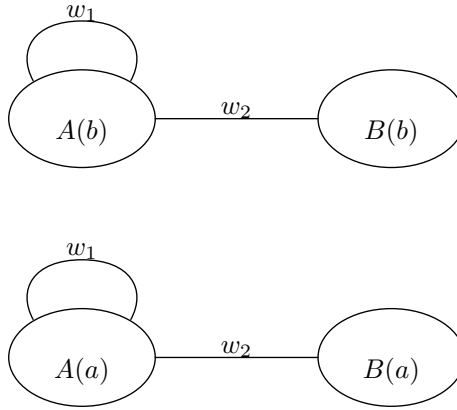


Fig. 2. Ground network for an MLN example containing a predicate dependency within individuals. For details see text.

scenario this would mean that the action of eating an apple by individual a can cause, with some probability, the expulsion from paradise for individual b or vice versa. This is reflected by a slightly amended version of our theory which looks like this

$$\begin{array}{l} w_1 \quad A(x) \\ w_2 \quad A(x) \Rightarrow B(y) \end{array}$$

Note that this change does not alter the set of possible worlds, but it alters the layout of the ground MRF as can be verified from Fig. 4. The key effect of this change is on the marginal probability $\Pr\{B(a)\}$ that has substantially increased (see results in table 3b compared to 1b). The obvious reason for this is that we now have more paths with positive weight influencing the truth of $B(a)$ in a given possible world. As these paths have considerably higher weight than the loops weighted w_1 connecting $A(\cdot)$ with itself, another overall effect is a decrease of the marginal probability $\Pr\{A(a)\}$ (same of course for $\Pr\{A(b)\}$) as can be verified from tables 1a vs 3a.

Finally, to examine the influence of a theorem assumed to hold with certainty, let us assume $x \neq y \Rightarrow (A(x) \Rightarrow \neg A(y))$. In our domain $\mathcal{D} = \{a, b\}$, this translates to an exclusive-or (XOR) constraint $(A(a) \wedge \neg A(b)) \vee (A(b) \wedge \neg A(a))$, which in our scenario could correspond to only one apple being available (and our first humans unwilling to share an apple they are eating, which is not what happened according to the book of Genesis). This change in fact alters the set of possible worlds without changing the layout of the MRF, because we are now forbidding all possible worlds in which the hard XOR constraint as formulated above does not hold. A visualization of the added constraint is provided as connector with dotted head and tail in Fig. 6.

The effect of this change to the probability distribution on the ground MRF can be conveniently described (and, in fact, coded) as *conditioning* on the set of

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|--------------------------|
| T | T | T | T | 8.6 |
| T | F | T | T | 8.3 |
| T | T | F | T | 4.6 |
| T | F | F | T | 6.3 |
| T | T | T | F | 4.6 |
| T | F | T | F | 6.3 |
| T | T | F | F | 0.6 |
| T | F | F | F | 4.3 |
| $Z = 27929.7$ | | | | $\Pr\{A(a)\} = 0.387371$ |

(a) Case I

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|--------------------------|
| T | T | T | T | 8.6 |
| F | T | T | T | 8.3 |
| T | F | T | T | 8.3 |
| F | F | T | T | 8. |
| T | T | T | F | 4.6 |
| F | T | T | F | 6.3 |
| T | F | T | F | 6.3 |
| F | F | T | F | 8. |
| $Z = 27929.7$ | | | | $\Pr\{B(a)\} = 0.738638$ |

(b) Case II

Fig. 3. Marginal probability example results for ground network from Fig 4

possible worlds compatible with the hard XOR constraint. Looking at the tables in Figure 5, it can be seen that we have only half as many satisfying possible worlds for our ground formulas as in the earlier versions of the example. The overall effect on the marginal probabilities for our test atoms is that we find another increase of $\Pr\{B(a)\}$ (see results in table 5b). In addition, there is a change of $\Pr\{A(a)\}$ to 0.5, which seems surprising at first glance. The simple explanation is that our XOR constraint changes the overall σ -algebra on our configuration space such that events $\{A(a)\}$ and $\{A(b)\}$ are now mutually exclusive and together exhaustive. Moreover, as $\Pr\{A(a)\} == \Pr\{A(b)\}$ must hold, the result in table 5a is justified.

Summing up, this example demonstrates a few important observations. First, there is no context-independent precise “meaning” of a weight in a MLN. Rather, these weights can be seen as ordinal constraints on theorem probabilities with different numeric interpretations depending on the interpretation context (domain data). Second, hard constraints can alter marginal probabilities in unexpected ways as they influence the overall configuration space of the ground MRF. Finally, on the positive side, MLNs can very easily handle referential uncertainty. Notice that our results were derived assuming trivial data only (no fixed assignment of any domain member to any predicate was assumed). This flexibility is of great importance in more complex problems (e.g. rules describing probable relationships in a social network graph).

3 Bayesian Logic Network (BLN) Concepts

Markov logic networks use *material implication*, usually denoted \Rightarrow , to express dependencies between predicates. Note that a material implication is true for any instance in which the antecedent (or IF part) is false, which differs somewhat from common intuition since an implication could seem to be simply not applicable to a case in which the IF part is false. This leads to a slightly unexpected behaviour of material implication if applied to statements with relative

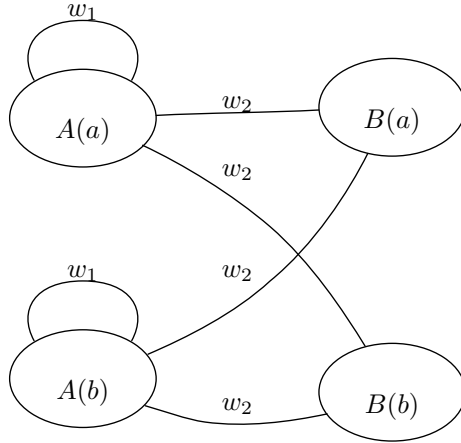


Fig. 4. Ground network for the modified example with a predicate dependency within and across individuals.

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|---------------------|
| T | F | T | T | 8.3 |
| T | F | F | T | 6.3 |
| T | F | T | F | 6.3 |
| T | F | F | F | 4.3 |
| $Z = 10373.4$ | | | | $\Pr\{A(a)\} = 0.5$ |

(a) Case I

| $A(a)$ | $A(b)$ | $B(a)$ | $B(b)$ | SumWts |
|---------------|--------|--------|--------|--------------------------|
| F | T | T | T | 8.3 |
| T | F | T | T | 8.3 |
| F | T | T | F | 6.3 |
| T | F | T | F | 6.3 |
| $Z = 10373.4$ | | | | $\Pr\{B(a)\} = 0.880797$ |

(b) Case II

Fig. 5. Marginal probability example results for ground network from Fig 6

frequencies. E.g., the formula $dog(X) \Rightarrow barks(X)$ is true in a grounding for individuals comprising not only dogs, but also all other animals, evens humans etc. Therefore, in practice, many irrelevant instances contribute to verifying a material implication.

The alternative approach underlying Bayesian networks and their extensions, e.g., Bayesian logic networks, is based on *conditional probability*. A conditional probability table (CPT) fully specifies probabilities for all outcomes of a random variable in the THEN part, given all possible combinations of values of the random variables in the IF part. Thus, there are never irrelevant cases in grounding variables being related in a CPT. E.g. the conditional dependency table for a dependency of $\Pr(barks(X))$ on $dog(X)$ contains an entry $\Pr(barks(X)|\neg dog(X))$ which requires an estimate of the probability that non-dogs would ever bark – a quantity that relates to the specificity of the dependency in question.

The obvious corrective action to take in specifying the input theory to a MLN is to add a formula with negated IF part to the input theory and assign

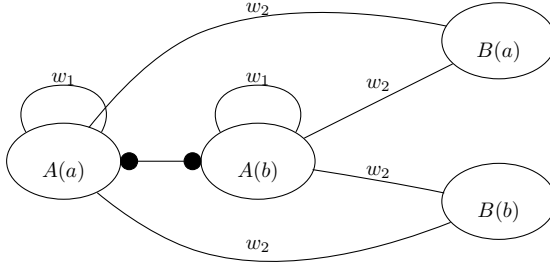


Fig. 6. Ground network for the modified example with an XOR constraint added for predicate $A(\cdot)$

it an appropriate weight. In the example above, it would make sense to add $\neg dog(X) \Rightarrow barks(X)$ with a rather high negative weight.

A further advantage of using conditional probability can be seen in the easier representation of dependencies involving discrete variables with multiple values. In such cases, an MLN input theory would require an auxiliary predicate for each such value and additional axioms asserting mutual exclusion and collective exhaustiveness of the auxiliary predicates for one multi-valued variable.

Besides this impracticability there is another, theoretical reason for distinguished treatment of conditional probability and the underlying dependency concept from the kind of dependence expressed in material implication. This theoretical reason has been investigated thoroughly in an approach related to so-called conditional objects. Here, it was proven that there is no object in a Boolean algebra that would correspond to the computation rule underlying the definition of conditional probability. Instead, it was shown that there is an entire range of sets and a Boolean algebra among which one can be chosen as representing a conditional [35]. This result, which links probabilistic logic to some theorems and fuzzy logic, has led to the concept of conditional events which has seen some applications in probabilistic expert systems [12]. For an application of this approach to evidential reasoning, see [48].

The Bayesian logic network is a framework for defining Bayesian networks using predicates and connectives in 1st order logic, with the specific assumption that material implication is to be replaced by probabilistic conditioning. Thus, a Bayesian logic network defines some predicates with finite valued domains. Allowed logical connectives between these predicates are negation, conjunction, and disjunction. However, it should be noted that the usual material implication, which can be written as a combination of disjunction and negation, will not be interpreted in a Bayesian logic network in the same way as it would in the Markov logic network. Rather, the conditioning operator, usually written as $|$, is used to indicate value dependencies between domains of predicates.

Therefore, in a Bayesian logic network, we have a grounding algorithm that is similar, but not equivalent to grounding in the Markov logic network. In the Bayesian logic network case, the key purpose of the grounding algorithm is a combination of 1st order logic entailment with proper grounding of the predicates involved.

As a consequence, the entailment and grounding computation steps lead to a standard Bayesian network. This includes the application of so-called combination rules to input theories that contain multiple dependency assertions with identical consequent variables. As algorithms for these networks are available in many implementations, the remainder of this paper will focus on Markov logic networks. For more details on the BLN approach, the reader is referred to [18] and [21] building on the ground breaking work [14].

4 Definition Languages and Inputs to a MLN Engine

In order to get an MLN engine to work for you, three kinds of input are needed. First, there is the input theory as discussed in section 2.1. Second, the current domain data are needed on the basis of which the MLN engine will compute posterior probabilities of ground atoms or a maximally probable posterior possible world (see section 5.2). Following the tradition of Bayesian networks, this input data is commonly referred to as *evidence*. Finally, in order to enable a focussed computation of posterior probabilities, the input to an MLN engine should specify queries relating to specific statements of interest. We will now summarize the conventions used for each of these inputs.

4.1 Input Theories

At least two approaches to defining input theories for a Markov Logic Network exist, namely a generalization in [18] building on the Bayesian Logic Language [32], and the TUFFY input file definitions [8] extending the syntax proposed in the ALCHEMY project [25]. For brevity, we dwell on the latter approach. Here, all input is provided in a single MLN input file with several sections. The basic inputs needed for constructing a MLN and for supporting the grounding process based on incorporation of evidence, are provided in the following sections of the MLN input file –

Predicate schemata – This section contains predicate names and arities declared by variable lists. Example $A(x), B(x), C(x, y), \dots$

Range restrictions – This section contains range definitions for predicates as far as desired by the knowledge engineer. Example $C(x, y) := A(x), B(y)$. constrains the first argument of predicate $C()$ to be of type $A()$ etc. Not restricting a range leads to generating all possible ground atoms over the cartesian product of predicate domains in the grounding phase.

Datalog rules – These rules are applied to the evidence files (see next subsection) in order to generate inferred tuples. The notation follows PROLOG language conventions (except for using predicate symbols starting with an uppercase letter, and variable symbols beginning with a lowercase letter, example $PossibleFriends(x, y) :- Human(x), Human(y), SameSportsClub(x, y)$.

These rules are basically helpers in order to ease evidence file writing.

MLN Inference rules – These are the proper input theory rules as discussed in section 2.1. Rules are either prefixed with a weight to indicate a degree of (positive or negative) prior certainty / confidence, or, alternatively, suffixed with a dot to indicate certainty.

There are a few additional syntactic constructs allowed in 8. The most important such construct allows to add constraints to datalog and inference rules with the effect of reducing the size of the ground Markov network as far as possible. An example is the constraint (in angle brackets)

PossibleFriends(x, y):- *Human*(x), *Human*(y), *SameSportsClub*(x, y), [$x \neq y$]

This constraint excludes the reflexive part from the friendship relationship in the given domain. A side remark regarding datalog rules – they could be stated as formulas holding with certainty in the input theory instead. The reason for allowing these rules is to move some straightforward inferencing out of the network grounding procedure (see section 5.1). However, future implementations of MLN might opt for an integrated inference procedure involving all possible inferential rules.

4.2 Input Evidence

An input evidence file consists of ground atoms only which must use predicates from the input theory. These evidential ground atoms usually are a proper subset of the Herbrand base of the theory and interpretation in question, so they specify a partial possible world. The MLN inference process will take the evidence to compute either posterior probabilities of ground atoms or a maximally probable posterior possible world using the inference process discussed in section 5.2.

4.3 Input Queries

Input queries are provided either as ground atoms only which use predicates from the input theory, or as predicate schemata with one or more variables. The meaning of the queries depends on the MLN inference mode – MAP or MPE, as explained in section 5.2. We provide a few simple examples for the MPE mode – query $A(b)$ requires the MLN to compute a posterior probability for atom $A(b)$ while query $C(a, x)$ assuming a to be a ground individual requests computation of all posterior probabilities for individuals in relationship $C(., .)$ to a , as far as they are elements of the range of the second argument of $C(., .)$. The effective range is computed using the range restrictions stated in the input theory file, the individuals explicitly assigned in the evidence file, and possible further individuals obtained from evaluating the datalog rules in the input theory file.

4.4 Relaxations of the Interpretation Assumptions

The assumption of unique names of constants is, of course, not relevant if the input theory has no constants which is true in most practical cases. Otherwise, a

good mechanism is to use scoped predicates in place of the constants and ensure that as many candidate atoms relating to the scoped predicates are generated as needed in the grounding process (by adding a suitable query relating to the predicate to the query file). The assumption of known individuals of each (one-place) predicate can be relaxed using a universal predicate (like `THING` in the web ontology language `OWL`) and assigning individuals to this predicate only. As a result, all legitimate substitutions for variables in one-place predicates will appear in atoms as part of the resulting ground MRF. Analogous constructions can be applied using a universal n -place relation for predicates involving more than one variable. The assumption of known functionally dependent individuals can be relaxed using an equality-like predicate and allowing suitable atoms on the candidate set for the domain of a functional relationship.

5 Computation Steps and Algorithms

The key computation steps in an MLN are

- the construction of a ground network given the input theory and interpretation data (domain, ground instances of some predicates as evidence),
- the processing of inference tasks on this ground network in order to generate end user output,
- (optionally) training data driven learning of formula weights or even of formula structure.

In the present section, we focus on ground network construction and inference processing.

5.1 Computation of Ground Networks

The prerequisite to computing a ground network is the availability of a set of named individuals corresponding or interpreting the variables and constants in the formal theory. The main task for Markov logic networks is to build a ground Markov network (Markov random field) from the given logical theory for the given named individuals and their functionally dependent further counterparts.

The challenge in building this ground network is to avoid as far as possible the exponential complexity involved in taking all possible combinations of predicates and possible argument ground terms. This can be done, first, by restricting grounding of formulas to individuals of proper types. Most available tools for MLN offer type declaration capabilities in the theory definition language, see section [4](#).

Second, the size of the ground network to be computed can be reduced significantly by integrating the grounding operation with computation of logical entailments using the clauses of the underlying theory Θ together with the evidence \mathcal{E} provided for the inference step (see next subsection). In a generic approach, an inference engine could be used to implement the computation of logical entailments, however, in both MLNs and BLNs the intention is to allow for a very

broad theory definition that may not conform to the restrictions imposed by, say, a PROLOG or ontology-based inference procedure. Therefore, the only restriction on Θ is satisfiability, and the grounding computation integrates entailments from the evidence in the course of the grounding process itself.

The TUFFY project [36] implements a grounding algorithm using an object-oriented relational database (specifically, PostGreSQL). Basically, TUFFY generates a table for each predicate containing a primary key, a sub-array with a field for each argument and a truth value in the range $\{T, F, U\}$, where U represents unknown values at the time of grounding. Using these tables, a table with all ground clauses can be constructed efficiently by performing appropriate join operations on these predicate tables and appending the weights from the input theory. It should be noted that in these computations the datalog rules as allowed in the TUFFY theory input files are taken into account, as well.

Additionally, the TUFFY project [36] exploits the fact that, in most cases, ground MRFs have many disconnected components.

5.2 Inference

The prerequisite to computing inference is a set of assumed statements, which correspond to ground atoms with truth values assigned. These statements are usually referred to as evidence \mathcal{E} or simply data. Usually, evidence does not fully specify a possible world so that many possible worlds are compatible with \mathcal{E} .

Inferencing Tasks. The main purpose of inference for Markov and Bayesian logic networks is to derive ground atoms with high posterior probability given the evidence. The computation of inference may be driven by queries that set specific goals in terms of ground atoms to be supported or refuted.

The most common inferencing task over Markov logic networks is finding the Most Probable Explanation (MPE), i.e. finding a configuration of the ground Markov network with maximum probability. This task is of particular relevance in absorbing evidence, since then an MPE solution provides the most probable overall state of affairs (or, in logical terms, possible world). For a formal definition of MPE, see [7], Def. 18.

While an MPE solution yields a full possible world, the other approach to inferencing focusses on posterior probabilities of specific hypothesis nodes. The term commonly used in Bayesian / Markov network literature for this kind of inference is maximum a posteriori (MAP, which is slightly misleading since MAP inference in the usual sense is defined in Bayesian statistics and presumes an optimization procedure involving prior distributions of parameters). In our setting, this kind of inference would correspond to weight learning, a subject we will address in section 5.3. If parameter learning is not involved, computing a MAP inference actually reduces to a marginalization operation performed for a specific set of query nodes in the ground network. In the TUFFY software, MPE inference is the default, which is referred to as MAP inference in the manual. On the other hand, the marginalization inference as just introduced is appropriately referred to as marginal mode and can be requested with a command line option.

Computational Approaches to Inferencing. There are two approaches to computing inference –

exact methods – these methods allow for the exact computation of posterior probabilities. Such methods are commonly subsumed under the term belief propagation. The two common approaches for exact inference or belief propagation are based on *message passing* or, alternatively, the *join tree algorithm*. Message passing, in an early version suggested in [37] and extended in the sum-product algorithm, [26], is an asynchronous scheme for weight aggregation across network neighbours in a specially constructed network, the factor graph. A factor graph for ground MLN is a bipartite graph with atom nodes in one partition, the clause cliques (factors) in another partition and an undirected link between each atom and the clauses it appears in. The corresponding algorithm is guaranteed to converge to exact posterior probabilities in two steps if acyclicity conditions are met on the factor graph. The complexity of the message passing algorithm under this assumption is linear in the size of the factor graph and the maximal number of states in a factor (clique) node. – The join tree algorithm was proposed originally by [28] and builds on a tree constructed by joining neighbouring cliques in the MRF following a given node ordering. Inference through this join tree proceeds in two steps by aggregating potentials upwards and then propagating normalization vectors downwards. The node ordering is derived by maximum cardinality search, where a root node / clique can be chosen depending on the direction of evidence processing. The performance of the join tree algorithm depends on the cut set sizes of adjacent nodes in the MRF. Ordering the nodes such as to optimize the cut sets to be visited in the resulting computation paths across the tree leads to significant improvements as proposed in the bucket-elimination approach by [19]. This is an important enhancement as the complexity of the entire join tree algorithm is exponential in the size of these clique joins.

stochastic methods – these methods use Markov Chain Monte Carlo (MCMC) approaches for stochastic relaxation in a network to derive approximate posterior probabilities or likelihoods. A very popular approach for Markov random fields is Gibbs sampling, see, e.g., [30], section 29.5. Gibbs sampling assumes that the MRF has a stationary distribution, which presupposes strict positivity of the configuration distribution and aperiodic reachability of each configuration from any other configuration [22]. Under these assumptions, Gibbs sampling is a method for approximating the marginal stationary distribution of node states. Procedurally, Gibbs sampling in an MRF consists of sampling states of nodes given their immediate network neighbours (commonly referred to as their Markov blanket). This amounts to estimating the conditional distribution at nodes being sampled which is proportional to the desired marginal distribution.

For MLN, as atom nodes may participate in many clauses, and as subclauses may appear multiple times in more complex clauses, it is usually the case that exact methods are not appropriate. The acyclicity of the factor graph cannot be

guaranteed, which leads to possibly long convergence times of the factor graph message passing algorithm. For the same reasons, the maximal size of the state set of a clique join can become very high, which makes the join tree algorithm hard to apply.

However, stochastic methods like Gibbs sampling cannot be applied to MLN inference in a straightforward way, since the key assumptions guaranteeing the existence of a stationary distribution in the ground MRF are not satisfied in general (for a derivation, see [2]). First, not all possible worlds have strictly positive probability. This is true since clauses assumed to be false with certainty are allowed in an MLN definition (as we illustrated in the example section). Second, not all possible worlds are reachable from any given one as some of them usually are in the set of unsatisfiable worlds given some partially defined possible world (either by evidence or in the course of sampling from the MRF).

These problems have been described in detail and addressed in the most common MCMC approach to MLN inference called MC-SAT [38]. Basically, MC-SAT builds on recent advances in stochastic satisfiability search (the SAT part) and combines these with an extension of Gibbs sampling appropriate for handling clear or approximate violations of strict positivity in the configuration distributions (the MC part).

As for the SAT part, MC-SAT approach to performing inference on a ground MRF builds on randomized algorithms for satisfiability solving (SAT-solvers). While satisfiability search for clauses involving at least 3 predicates is a classical NP-complete problem [17], the combination of a random walk with deterministic steps of clause or atom addition is applicable in most cases and was published as the original WalkSAT algorithm [41]. WalkSAT is a combinatorial optimization heuristic which uses the number of overall satisfied clauses as objective function. WalkSAT interleaves greedy steps adding compatible atoms to the possible world being constructed with random walk steps – “flips” of randomly selected clause – that enable exploring other possible worlds and escaping from local maxima of the objective function. WalkSAT led to a breakthrough in many applications that require SAT solving, eg in constraint processing and systems verification. In [51] it was shown that WalkSAT samples in a highly non-uniform way and repeatedly finds satisfying configurations with high similarity. This led to the proposal of the SampleSAT algorithm, which adds simulated annealing steps to WalkSAT in order to enable moves to highly uncorrelated partial possible worlds during the search process. In many empirical demonstrations, the highly efficient behaviour of SampleSAT was confirmed. Today, SampleSAT is usually combined with additional heuristics like taboo search, for details and software, the reader is invited to check [1].

As for the MC part, the approach in [38] implements an adapted version of slice sampling [34]. This is a comparatively new method to sampling from probability distributions in high dimensional spaces. Slice sampling uses an auxiliary variable taking values in the range of the probability density function (or a function proportional to it). It then iterates over sampling the aux variable and the original sample space in an alternating way such that samples from

the sample space must have a probability (or likelihood) at least equal to the current sampled value of the aux variable. Both sampling steps use iteratively re-parameterized uniform distributions. To adapt this approach to the needs of ground MRF processing, several steps are proposed in [38] –

- First, the input clauses with negative weights are transformed such that all weights are non-negative. Note that any clause that is part of a conjunctive normal form is a disjunction, so that negating this clause amounts to generating a set of conjunctions. The components of these conjunctions will appear with equal weights in the transformed ground MRF. As a result of this transformation, all potentials in the ground MRF are non-negative. This, in turn, allows to sample potentials from a uniform distribution in an interval $[0, a]$, which is needed in the next step.
- The second step implements the sampling step from the distribution function similar to [34] to the transformed MRF. In [38], it is proposed to use auxiliary variables added to each clique $1, \dots, k, \dots$ in the transformed ground MRF. The values of these per-clique variables u_k are sampled uniformly in each MRF update step from the range of clique likelihoods. As a result, a random subset of candidate next configurations is produced whose probabilities lie close to the probability of a possible world with the ground formulas as satisfied in the current configuration \mathbf{p} . However, the candidate configurations may have only small overlaps with \mathbf{p} in terms of ground clauses satisfied.
- The third and last step is to to apply SampleSAT to the candidate set established in the former step. This approximates a selection of the next configuration with uniform distribution as required in slice sampling.

The overall strategy of the MC-SAT inference algorithm is to use the MC part for establishing probable configurations and, alternating with it, the SAT part for selecting a feasible configuration under the constraints of the input logical theory. MC-SAT [38] has been demonstrated in many simulations and applications to be highly efficient in comparison to any other approaches tested, and it is adopted in both the Alchemy [25] and TUFFY [36] MLN engines.

An additional option for building more effective data structures in inference computing on ground MRFs for MLNs exploits the underlying logical dependencies across the network nodes. This is being investigated as *lifted* inference. One recent approach [9] is based on pushing the propagation of logical constraints into the construction of an AND-OR search graph. This graph is then simplified in a second step by aggregating nodes with equal weights. An earlier approach [43], implemented in the Alchemy system, builds on a message-passing implementation of belief propagation and uses an iterative approach to alternating updates of the weights in ground clauses and in ground predicates (where the weights here represent potentials).

5.3 Learning

In both Bayesian and Markov logic networks, learning builds on multi-relational data. For Bayesian networks, we have a standard distinction between structure

and parameter learning. Structure learning is known to be NP - hard. Parameter learning can be performed based on complete and incomplete data using likelihood maximization.

The main purpose of learning for Markov and Bayesian logic networks is to select or derive clauses with high posterior probability given the training data. *Selection* of likely clauses in the light of training data is usually referred to as *weight learning*, and in MLN it corresponds to adjusting the weights w_i of a given set of clauses F_i as defined in section 2.1. From a statistical point of view, weight learning in a Markov random field is nothing but parameter estimation. *Deriving* likely clauses given training data is usually referred to as *structure learning*. Structure learning assumes an additional step of generating appropriate clause candidates. This can be done, e.g., by combining inductive logical programming (ILP) methods with a parameter estimation algorithm.

Weight Learning. The challenge of *weight learning* in Markov logic networks is the complexity of the likelihood function, since computing this function usually involves an exponential number of ground atoms to be considered. Therefore, straightforward simulation approaches like the ones conventionally used in weight learning for Boltzmann machines [450] are not scalable enough for the job. The approach adopted in the Alchemy and TUFFY prototypes builds on [29]. This paper combines several advances beyond the state of the art in numerical optimization and parameter estimation. Given the tutorial nature of this paper, we only briefly summarize the two most important aspects –

Scalability of Objective Function– Standard MRF parameter estimation proceeds by iteratively minimizing an objective function. The usual objective function in MRF parameter estimation is the Kullback-Leibler divergence measure that evaluates proximity (in a general sense, not equivalent to a metric in the strict mathematical sense) of two probability distributions. In the case of MRF, the proximity is computed for the vector of unconditional local distributions of node states in the entire MRF versus the conditional distribution vector given the data (imagined as “clamping” nodes in the network whose states are fixed by the evidence). It has been shown that minimizing Kullback-Leibler divergence is equivalent to maximizing the conditional log-likelihood of the data under the MRF parameters [50]. The issue with computing this function is that approximating the unconditional distribution requires running the Markov chain for a sufficiently large number of iterations which is usual exponential in the size of the network. Sampling the conditional distributions is less complex, in the case of full evidence a single inference step or belief propagation step suffices per data sample. A more efficient standard of comparison summarizing the unconditional probability was suggested in [16] under the term *contrastive divergence*. The approach here is to compute just one update of the MRF configuration in the unconditional state, which implies calling a random sequence of node updates using Gibbs sampling as described above once for each node. This gives an estimated local probability vector for the unconditioned MRF. Computation of

the conditioned local probabilities remains unchanged. In [16] it was shown that contrastive divergence is far more efficient and at least as effective as the conventional Kullback-Leibler divergence based procedure.

Flexible and Efficient Weight Adaptation Scheme— The iterative adaptation of connectivity weights based on evaluating the objective function needs defining step directions and step sizes for each connection weight change. While, in principle, step directions are given for a divergence measure by its gradient, it has been shown that straight forward gradient descent is not optimally performant and can be replaced by suitable variants of *conjugate gradient descent*. In its general form, a conjugate gradient descent computation of a solution to an inhomogeneous system of linear equations ensures orthogonality of each update vector with the earlier update vectors under the mapping of the constraint matrix (for details, see [42]). This principle has been suitably modified in [29] to be applied to MLN weight learning.

The overall weight learning algorithm proposed in [29] also uses the inference algorithm MC-SAT [38] as explained in section 5.2 for moving to probable successive configurations in the ground MRF state space. Compared to standard Gibbs sampling, the two advantages gained by the intervening SampleSAT step are that logical constraints are being taken into account preventing search from moving through unfeasible regions and that successively visited states may have very low correlation. For further aspects and details of the weight learning approach followed in the *Alchemy* and *TUFFY* prototypes, the reader should consult [29].

Structure Learning. The currently most promising approach to structure learning [24] builds on the hypergraph representation of a relational database schema. In this representation, each attribute defined in the schema corresponds to a node, and a hyperedge is introduced for the set of attributes of each table in the database. Relational data corresponding to a given schema are assumed to be available as learning data. The learning algorithm now picks attributes and rearranges the hyperedges by a cluster-finding approach in order to establish weighted implication rules combining these new hyperedges. Clustering and rule mining are defined such as to maximize the likelihood of the given learning data.

6 Overview of Applications

Applications of MLNs are mostly attractive in areas where some degree of a priori knowledge is available, and where multi-relational data in sufficient size is available. In particular, knowledge expressible as rules including usual rules and heuristic rules is of interest for a MLN application. One area with specific appeal for the MLN approach is information management, since in almost all areas here we have expert rules available that are not true in all cases. This holds for tasks like topic identification, named-entity recognition, parsing in shallow natural language processing etc. As an example application, we mention the so-called *machine reading* prototype [39], which performs automated extraction of

information from large bodies of textual data (as available from web information resources including blogs, newsgroups, social networks). Here, information extraction means entity recognition and resolution, relationship detection and verification, and classification of entities or relationships. The broader perspective for solutions in machine reading is to enable automated problem oriented text understanding, as it is currently demonstrated by the IBM Watson software and its capabilities in answering jeopardy questions.

For BLN applications, multi-entity dependencies optionally involving constraints are important, as they appear in genomic studies [21].

An important field for applications is *Operational Risk Management*, see [46] and foundations in [47]. The benefits of a predicate-logic approach are mainly due to the complex domain models needed in this field, which usually require representation of operational systems, operative processes, client or field systems, and service deployments based on service contracts. In [46], based on the EU FP6 MUSING integrated project, the domain model was expressed in a formal ontology using the web ontology language OWL. Ground networks were generated on instantiated ontologies using the Jena rules language and appropriate inferencing systems. A prototype was deployed at an IT operations center of a MUSING industry partner.

Another highly promising field for MLNs and/or BLNs is *Information Extraction and Semantic Parsing*, see [23] and [39]. The flexibility of MLNs in representing relational data with very generic associations in combination with a rigorous predicate logic approach is highly advantageous here.

7 Outlook and Conclusion

Overall, both MLN and BLN approaches are highly promising approaches for addressing predicate-logic based uncertainty in practical applications. Algorithms addressing all key computational challenges are available. Practical examples are available and are being tested on data sets of increasingly practically meaningful sizes.

Technically, an open issue is the relationship of MLN definition languages to ontologies based on the OWL 2 sub-languages. The strict limitations of expressivity imposed in OWL 2 dialects might help to select input theories of known inferential complexity, and this might benefit inference and learning processes. – Another open issue is the possible use of other logic programming approaches from answer set programming for the network grounding computations.

Tools for MLN are available as mentioned throughout the paper. We hope that, as a result from this tutorial, students and researchers will explore further applications and refinements of these promising approaches.

Acknowledgement. The author is indebted to three anonymous referees who gave very specific and helpful comments. I would also like to thank the organizers of the EBISS 2012 event in Brussels, my colleagues Marie-Aude Aaufaure and Esteban Zimanyi, for their support. Finally, the audience at the EBISS 2012 helped to improve some parts of the paper by discussions and interesting questions.

References

1. <http://www.cs.rochester.edu/u/kautz/walksat/>
2. Aarts, E., Korst, J.: *Simulated Annealing and Boltzmann Machines*. Wiley (1988)
3. Aarts, E., Korst, J.: Computations in massively parallel networks based on the boltzmann machine: A review. *Parallel Computing* 6, 129–145 (1989)
4. Ackley, D., Hinton, G., Sejnowski, T.: A learning algorithm for boltzmann machines. *Cognitive Science* 9, 147–169 (1985)
5. Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.: *The Description Logic Handbook - Theory, Implementation and Algorithms*. Cambridge University Press, Cambridge (2004)
6. Borgelt, C., Kruse, R.: *Graphical Models: Methods for Data Analysis and Mining*. Wiley (2002)
7. Dechter, R., Mateescu, R.: And/or search spaces for graphical models. *Artif. Intell.* 171(2-3), 73–106 (2007),
<http://dx.doi.org/10.1016/j.artint.2006.11.003>
8. Doan, A., Niu, F., Ré, C., Shavlik, J.: User manual of tuffy 0.3. Tech. rep., University of Wisconsin-Madison (2011)
9. Domingos, P., Gogate, V.: Exploiting logical structure in lifted probabilistic inference (2010), <http://ai.cs.washington.edu/pubs/204>
10. Domingos, P., Richardson, M.: Markov Logic: A Unifying Framework for Statistical Relational Learning. In: Getoor, Taskar (eds.) [11], ch. 12, pp. 339–372 (2007)
11. Getoor, L., Taskar, B. (eds.): *Introduction to Statistical Relational Learning*. MIT Press, Cambridge (2007)
12. Goodman, I., Nguyen, H., Rogers, G., Gupta, M.: *Conditional Logic in Expert Systems*. North Holland, Amsterdam (1990)
13. Heckerman, D.: A tutorial on learning with bayesian networks. Tech. rep., Microsoft Research, Redmond, Washington (1995)
14. Heckerman, D., Meck, C., Koller, D.: Probabilistic Entity-Relationship Models, PRMs, and Plate Models. In: Getoor, Taskar (eds.) [11], pp. 201–238 (2007)
15. Hintikka, J.: *Knowledge and the Known*. Synthese Historical Library. D. Reidel Publishing Company, Dordrecht (1974)
16. Hinton, G.: Training products of experts by minimizing contrastive divergence. *Neural Computation* 14, 2002 (2000)
17. Hopcroft, J., Motwani, R., Ullman, J.: *Introduction to Automata Theory, Languages and Computation*. Pearson, Upper Saddle River (2003)
18. Jain, D., Waldherr, S., Beetz, M.: Bayesian logic networks (extended version, included in probcog tool distribution). Tech. rep., TU München (2011), <http://wwwbeetz.informatik.tu-muenchen.de/probcog-wiki/index.php>
19. Kask, K., Dechter, R., Larrosa, J., Dechter, A.: Unifying cluster-tree decompositions for reasoning in graphical models. *Artificial Intelligence* 166, 165–193 (2005)
20. Kern-Isberner, G., Beierle, C., Finthammer, M., Thimm, M.: Probabilistic Logics in Expert Systems: Approaches, Implementations, and Applications. In: Hameurlain, A., Liddle, S.W., Schewe, K.-D., Zhou, X. (eds.) *DEXA 2011, Part I. LNCS*, vol. 6860, pp. 27–46. Springer, Heidelberg (2011)
21. Kersting, K., De Raedt, L.: Bayesian Logic Programming: Theory and Tool. In: Getoor, Taskar (eds.) [11], pp. 291–322 (2007)
22. Kindermann, R., Snell, J.L.: *Markov Random Fields and their Applications*. American Mathematical Society (1980)

23. Kok, S., Domingos, P.: Extracting semantic networks from text via relational clustering. Tech. rep., Department of Computer Science and Engineering, University of Washington (2009)
24. Kok, S., Domingos, P.: Learning markov logic network structure via hypergraph lifting. In: Proceedings of the 26th International Conference on Machine Learning (2009)
25. Kok, S., Singla, P., Richardson, M., Domingos, P.: The alchemy system for statistical relational AI (2005), <http://www.cs.washington.edu/ai/alchemy>
26. Kschischang, F., Frey, B.J., Loeliger, H.A.: Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory* 47(2), 498–519 (2001)
27. Laskey, K.: Mebn: A logic for open-world probabilistic reasoning. Tech. rep., George Mason University (2006)
28. Lauritzen, S., Spiegelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. *J. R. Statistical Society B* 50(2), 157–224 (1988)
29. Lowd, D., Domingos, P.: Efficient Weight Learning for Markov Logic Networks. In: Kok, J.N., Koronacki, J., Lopez de Mantaras, R., Matwin, S., Mladenič, D., Skowron, A. (eds.) PKDD 2007. LNCS (LNAI), vol. 4702, pp. 200–211. Springer, Heidelberg (2007)
30. McKay, D.: *Information Theory, Inference, and Learning Algorithms*. Cambridge University Press (2003)
31. Mendelson, E.: *Introduction to Mathematical Logic*. Chapman Hall, London (1997)
32. Milch, B., Martha, B., Russell, S., Sontag, D., Ong, D., Kolobov, A.: BLOG: Probabilistic Models with Unknown Objects. In: Getoor, Taskar (eds.) [11], ch. 13, pp. 373–398 (2007)
33. Motik, B., Patel-Schneider, P., Parsia, B.: Owl 2 web ontology language structural specification and functional-style syntax (2009)
34. Neal, R.M.: Slice sampling source. *Ann. Statist.* 31(3), 705–767 (2003)
35. Nguyen, H., Rogers, G.: Conditioning Operators in a Logic of Conditionals. In: Goodman (ed.) [12], pp. 159–180 (1990)
36. Niu, F., Ré, C., Doan, A., Shavlik, J.: Tuffy: Scaling up statistical inference in markov logic networks using an rdbms. In: Proceedings of the VLDB Endowment, vol. 4. VLDB Endowment (2011)
37. Pearl, J.: *Probabilistic Reasoning in intelligent Systems: Networks of Plausible Inference*. Morgan Kaufman, San Mateo (1988)
38. Poon, H., Domingos, P.: Sound and efficient inference with probabilistic and deterministic dependencies (2006), <http://ai.cs.washington.edu/www/media/papers/poon06.pdf>
39. Poon, H., Domingos, P.: Machine reading: A “killer app” for statistical relational AI. In: Proc. AAAI Conference (2010)
40. Richardson, M., Domingos, P.: Markov logic networks. *Mach. Learn.* 62(1-2), 107–136 (2006)
41. Selman, B., Kautz, H., Cohen, B.: Local search strategies for satisfiability testing. In: Johnson, D.S., Trick, M.A. (eds.) *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge*, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 26. American Mathematical Society (1993)
42. Shewchuk, J.R.: An introduction to the conjugate gradient method without the agonizing pain. Tech. rep., School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213 (1994), <http://www.cs.cmu.edu/~quake-papers/painless-conjugate-gradient.pdf>

43. Singla, P., Domingos, P.: Lifted first-order belief propagation. In: Proc. AAAI Conference, pp. 1094–1099. AAAI (2008)
44. Smullyan, R.: First Order Logic. Dover, New York (1995)
45. Spies, M.: Einführung in die Logik - Werkzeuge für Wissensrepräsentation und Wissensmanagement. Spektrum Akademischer Verlag, Heidelberg (2004)
46. Spies, M.: Probabilistic Relational Models for Operational Risk: A New Application Area and an Implementation Using Domain Ontologies. Studies in Theoretical and Applied Statistics, pp. 385–395. Springer, Heidelberg (2011)
47. Spies, M., Schacher, M., Gubser, R.: Intelligent Regulatory Compliance, ch. 12, pp. 215–238. Wiley, New York (2010)
48. Spies, M.: Conditional events, conditioning and random sets. IEEE Transaction on Systems, Man and Cybernetics 24(12), 1755–1763 (1994), beitr
49. Spies, M.: Das langzeitgedächtnis als boltzmann maschine – eine simulation mentaler datenmodelle. Kognitionswissenschaft 8(2), 49–73 (1999)
50. Spies, M.: Contextual Learning and Retrieval in a stochastic Network, vol. 2, pp. 1943–1950. ESIA-Universität de Savoie (2002)
51. Wei, W., Erenrich, J., Selman, B.: Towards efficient sampling: exploiting random walk strategies. In: Proceedings of the 19th National Conference on Artificial Intelligence, AAAI 2004, pp. 670–676. AAAI Press (2004), <http://dl.acm.org/citation.cfm?id=1597148.1597256>

Large Graph Mining: Recent Developments, Challenges and Potential Solutions

Sabri Skhiri and Salim Jouili

Euranova R&D

Rue Emile Francqui, 4, 1435 Mont-St-Guibert, Belgium

{sabri.skhiri, salim.jouili}@euranova.eu

<http://www.euranova.eu>

Abstract. With the recent growth of the graph-based data, the large graph processing becomes more and more important. In order to explore and to extract knowledge from such data, graph mining methods, like community detection, is a necessity. Although the graph mining is a relatively recent development in the Data Mining domain, it has been studied extensively in different areas (biology, social networks, telecommunications and Internet). The legacy graph processing tools mainly rely on single machine computational capacity, which cannot process large graph with billions of nodes. Therefore, the main challenge of new tools and frameworks lies on the development of new paradigms that are scalable, efficient and flexible. In this paper, we will review the new paradigms of large graph processing and their applications to graph mining domain using the distributed and shared nothing approach used for large data by Internet players. The paper will be organized as a walk through different industrial needs in terms of graph mining passing by the existing solutions. Finally, we will expose a set of open research questions linked with several new business requirements as the graph data warehouse.

Keywords: Data Mining, Large graphs, Distributed Processing, Business Intelligence.

1 Introduction

Data mining is defined variously in the literature of computer science but the common use of the term corresponds to a process of discovering patterns or models for data. The patterns, however, often consist of previously unknown and implicit information and knowledge embedded within a data set [17]. That is, data mining is the process of analyzing data from different perspectives and summarizing it into useful information. In the literature, one can find a large scope of different methods and algorithms that deal with data mining, each of which has its own advantages and suitable application domains [24,31,44]. Those techniques have been heavily developed these last years in Business intelligence [41,50] especially for database and flat data in order to feed market analysis, business management, and assisted-decision tools [17]. It is worth saying that

the data mining stands at the intersection of different disciplines such as statistics, machine learning, information retrieval and pattern recognition. Almost all mining algorithms can be divided into the following families: (1) the classification for which we position data in predetermined groups, (2) clustering in which data are grouped within partitions according to different criteria, (3) associations that enables to link data between each other, (4) pattern recognition in which we mine data to retrieve predetermined pattern, (5) feature extraction and (6) Summarization (Ranking such as *PageRank*).

Traditionally, the algorithms manage and process the data as a collection of independent instances of a single relation. That is, the instances of data to be mined are considered independent without relationships between them. For example, in the case of the clustering algorithm in which the input data set is divided into groups with similar objects, it is considered that there is no relation between the objects. Hence almost all clustering algorithms compute the similarity between all the pair of objects in the data set by means of a distance measure. Indeed, the traditional data mining works are focused on multi-dimensional and text data. However, nowadays new emergent industrial needs lead to deal with structured, heterogeneous data instead of traditional multi-dimensional models. This kind of structured dataset is well designed as graph that models a set of objects that can be linked in a numerous ways. The greater expressive power of the graph encourages their use in extremely diverse domains. For example, in biology, the biochemical networks such as the metabolic pathways and the genetic regulation known as the transduction signal networks constitute a significant graph of interactions. On the other hand, the graphs are used in chemical data processing in a way that the molecule structure is described as a graph which implies that the molecule catalogs are processed as graph set. In the Internet area, the rise of social networks shown the need to model the social interactions as graphs. We can find another example in credit card fraud detection in which transactions are modeled as a bipartite graph of users and vendors.

This modeling change involves a paradigm shift in the way to apply the mining algorithms. We need to ensure that the classical data mining techniques are still equally applicable on graph models. The problem that arises here results from the fact that almost all needed measures such as similarity and distance cannot be easily defined for graph in as intuitive way as is the case for multidimensional data. As a matter of fact, the mining algorithms for graph are more challenging to implement because of the structural nature of the data. The second challenge that arises in many applications of graph mining is that the graphs are usually very large scale in nature. Indeed, in practice, those graphs can reach a significant size, as in social networks or interaction graphs. This kind of graphs can typically reach several hundred millions of nodes and billions of edges. However, most of the graph mining algorithms deal with data already available in main memory which is not accurate in large scale graph.

These last years we have seen new techniques emerging for facing this kind of large graph processing: (1) high performance graph database such as DEX [55] or Titan [64], (2) in-memory and HPC/MPI graph processing such as SNAP [5,6]

and finally (3) the distributed approach based on Bulk Synchronous Processing such as Pregel [51]. As a result graph miners will face three important issues: (1) adapting the mining algorithms to make them graph-aware, (2) redesigning the algorithms to be implemented by those new high performance techniques, (3) storing and exploiting many different graphs and to be able to apply similar processing as in traditional data warehouses.

In this paper we will focus on the distributed processing approach and we will show how this new generation of distributed graph processing frameworks can be used to implement typical graph mining algorithms. The second half of the paper describes whether having a high performance data mining stack is a sufficient condition to get a graph data warehouse stack.

Section 2 will present well-known data mining algorithms in clustering and classification areas. Section 3 will introduce new emerging distributed graph processing frameworks and will describe how the algorithms previously exposed can be implemented on such frameworks. Finally, Section 4 extends the concept of the graph mining to graph data warehouse processing and describes the new challenges that must be tackled by the research communities in order to reach the same level of performance as existing relational data warehouses.

2 Graph Mining Algorithms

The objective of this section is to introduce typical graph mining algorithms that will be discussed in the next sections for their distributed implementations on the Pregel paradigm. We first introduce a traditional graph mining algorithm used for large graphs, *PageRank*, and then we present a typical example of an existing mining algorithm for clustering that must be adapted to be graph-aware and fully leverage the linkage information.

2.1 Ranking: *PageRank*

The world wide web structure can be seen as a graph in which the web pages are the vertices and the (hyper-)links are the edges. However, in addition to the web page contents, the graph structure of the web presents a very important additional source of information which can hold implicit knowledge about the web pages. Since the nineties, some works have focalized their efforts on how to exploit this topological structure of world wide web (see, e.g., [11,12,14,17,20,42,43,54,59]). One of the most famous works which exploits the topological structure of the web is the *PageRank* algorithm [14,59]. This algorithm has been stated as one of the key to success of the well-known Google search engine [2].

The *PageRank* algorithm computes a ranking for every web page based only on the linkage structure of the world wide web (graph of the web). The authors of *PageRank* introduce the notion of page authority, which is independent of

¹ www.google.com

the page content. In *PageRank* algorithm, the authority is approximated from the number and importance of the pages pointing to the involved page. This algorithm considers a page as “important” if it has many incoming pages and/or it has a few highly ranked incoming pages. That is, a page has high authority (rank) if the sum of authorities (ranks) of its incoming pages is high.

The *PageRank* of pages is computed by following a random surfer which browses the web from page to page. The random surfer is a random walk such that the set of states is the set of Web graph vertices, and at each random step, with some probabilities, (1) the surfer chooses an outgoing link of the current vertex uniformly at random, and follow that link to the destination vertex, or (2) it “teleports”² to a completely random Web page, independent of the links out of the current vertex. Intuitively, the random surfer traverses frequently “important” vertices with many vertices pointing to it.

Let $\mathbb{G} = (V, E)$ be the web graph with vertex set V and edge set E . Let $\mathbf{d}_{out}(v)$ be the number of outgoing edges from the vertex $v \in V$. Let $\mathbf{d}_{in}(v)$ be the number of incoming edges to the vertex $v \in V$, i.e., the in-degree of v . Let p , ($0 < p < 1$), be the damping factor (usually set to 0.85) that represents the probability with which the surfer follows with the random walk, while $1 - p$ is the probability of teleporting to a random vertex among all $|V|$ vertices. Thus, the *PageRank* $\mathbf{PR}(v)$ of vertex (page) v is given by the following formula [59]:

$$\mathbf{PR}(v) = \frac{(1-p)}{|V|} + p \times \sum_{u \in \mathbf{d}_{in}(v)} \frac{\mathbf{PR}(u)}{\mathbf{d}_{out}(u)}$$

In a matrix form, Equation 2.1 can be rewritten as:

$$\mathbf{R} = p \times (\mathbf{A}\mathbf{R} + D) \quad (1)$$

where the matrix \mathbf{A} is a square matrix with the rows and columns corresponding to graph vertices, $\mathbf{A}_{u,v} = \frac{1}{\mathbf{d}_{out}(u)}$ if there is an edge from u to v and $\mathbf{A}_{u,v} = 0$ if not, \mathbf{R} is a column vector representing the ranks of pages, and D is a constant vector ($= (1-p)/|V|$).

While we will not go deeper into the mathematical underpinnings of *PageRank* here, it is shown that Equation 1 can be resolved with an iterative solution (see Equation 2) that converges for $0 < p < 1$:

$$\mathbf{R}_{i+1} = p \times (\mathbf{A}\mathbf{R}_i + D). \quad (2)$$

2.2 Graph Clustering

Clustering data is a fundamental task and one of the most studied topics in data mining [35,39]. Given a set of data instances, the goal is to group them into groups that share common characteristics based on similarity. Intuitively,

² *Teleportation* step: choose a vertex uniformly at random, and jump to it. This step is needed because it exists some vertices that does not have outgoing links (non-ergodic graph).

instances within a cluster are more similar to each other than they are to an instance belonging to different clusters. In the context of graph data, the clustering task is usually referred to as communities detection within graph [19,21,29,56,62,69]. In the case of a citation graph of the scientific literature, the vertices correspond to the papers and the edges correspond to citation relationships³. By clustering this graph, for a given paper one can identify the community of surrounding relevant works, without traversing the cited works nor the works they cite as well.

Here, we describe two graph clustering algorithms: the first algorithm is a generalization of the well-known k -means [50] algorithm and the second is a divisive algorithm which uses a structural-based index to gather information about the separable communities in a graph.

k -Means Based Clustering. In multidimensional data context, the k -means [50] algorithm is the simplest and one of the popular algorithms used for clustering. It minimizes the sum of the distances between the data instances and the corresponding centroids. The k -means [50] algorithm needs two parameters : (1) k the number of groups to provide and (2) $D : (o_i, o_j) \rightarrow \mathcal{R}$ a distance measure that maps pairs of data instances to a real value, it works as follows:

1. Randomly selects k data instances as the initial cluster centers (“centroids”).
2. Each data instance in the dataset is assigned to the nearest cluster, based on the distance (computed by D) between each one and each cluster center.
3. Each cluster center is recomputed as the average of the data instances in that cluster.
4. Steps 2 and 3 are repeated until the clusters converge. Convergence may be defined differently depending upon the implementation, but it normally means that either no objects change clusters when Steps 2 and 3 are repeated or that the changes do not make a material difference in the definition of the clusters.

The k -means algorithm has been extended recently to allow its use in linkage structure graph. In order to achieve reliable and efficient generalization of k -means to graph domain, two key issues have been addressed. Firstly, the distance measure D has been defined in such way it takes into account the relationship between vertices. Intuitively, the distance between two vertices is considered as the geodesic distance which is the number of edges (“hops”) in a shortest path connecting the pair of vertices in question. Secondly, the computation of the cluster centers (“centroids”) requires graph-aware procedures that can efficiently select a vertex which is the most representative of a set of vertices. One possible solution consists of using the notion of *median vertex* which is a vertex that minimizes the sum of distances to all the other vertices. Formally, let C be a set of vertices and D a distance measure for the graph, the median vertex \hat{v} of the set C is defined as follows:

³ Two papers are connected by an edge if one cites the other.

$$\hat{v} = \underset{v \in C}{\operatorname{argmin}} \sum_{u \in C} D(v, u). \quad (3)$$

Besides the median vertex, Rattigan et al. [62] uses the closeness centrality [27,72] to select the representative vertex of a cluster, i.e. they select the vertex with the greatest closeness score. A vertex will be considered in a central position according to his closeness score. This measure stresses that the quality (position in the graph) is more prominent than quantity (number of incident edges). Formally, the closeness centrality measure of a vertex v in a cluster C is computed as follows:

$$CC(v) = \frac{|V| - 1}{\sum_{v \neq u, u \in C} D(v, u)} \quad (4)$$

where $d(u, v)$ is a distance measure for the graph and $|V|$ the size of the graph.

As it can be remarked, one of the most important and common task in the graph clustering (and other graph algorithms) is the distance computation between vertex. As aforementioned, the shortest path computation forms the most used way to compute such distance. Intuitively, the goal is to find the shortest path from a source vertex v to a target vertex u , among all paths that satisfy a certain criterion. The applications of the shortest path computation cover a large scope of computer science fields, including network optimization [7,48], scheduling [28], image processing [57], geographic information systems [73], social network [75]. In the literature, since the late 1950's, shortest path problem was very well studied and many solutions to this problem have been proposed (i.e. see [68]). However, the earlier algorithms [9,22,26] are still used nowadays, especially Dijkstra's algorithm [22]. Dijkstra's algorithm solves the single-source shortest-path problem when all edges have nonnegative weights. It starts at a source vertex and explores the entire graph in all directions until the distances to all the other vertices (reachable from the source) are computed. Dijkstra's algorithm is considered as a greedy algorithm because it selects, in each step of the traversal process, the local optimum, which is the edge that satisfies the considered criterion.

Centrality Based Clustering. Before detailing this part, we provide a definition of one of the numerous centrality measures that is used for graph clustering. Here, we focus on the edge betweenness centrality [27,72] which locates, structurally (content-independent), the *well-connected* edges within a network. Here, an edge is considered to be well-connected if it is located on many shortest paths between pairs of vertices. That is, an edge with high betweenness centrality performs, in some sense, a control over the interactions between vertices. For instance, if two non-adjacent vertices v and w seek to communicate (interact) and their shortest path pass through an edge e , then e may have some control over the communication between v and w . This betweenness centrality of an edge e is calculated as follows:

$$BC(e) = \sum_{v, w \in V} \frac{b_{vw}(e)}{b_{vw}} \quad (5)$$

where $b_{vw}(e)$ is the number of shortest paths from v to w that pass through e and b_{vw} is the number of all shortest path between v and w .

Girvan and Newman [29] propose a divisive method that is well-suited for the social context. The divisive method starts with the whole graph and iteratively cuts specific edges. This divide the graph progressively into smaller and smaller disjoint subgraphs (communities). The key problem for these methods is the selection of the edges to be cut. Indeed, the edges to be cut should connect, as much as possible, vertices in different communities and not those within the same communities. Girvan and Newman [29] use edge betweenness centrality to select edges to be cut, for connected graphs. The idea behind this is that the inter-community edges have high betweenness centrality, while the intra-community edges have low betweenness centrality. The proposed algorithm works iteratively in two steps:

1. Compute the betweenness of all existing edges
2. Remove the edge with highest betweenness centrality
3. Repeat Steps 1 & 2 until the communities are suitably found

The stopping criteria can be designed by a “*a priori*” definition of “*suitable communities*”, and at each iteration we test whether the resulting subgraphs fulfill the definition. It is worth saying that this algorithm is very useful for web graph and social graph because they are characterized by small-world structure property [44,46].

Summarizing, it is clear that the graph clustering is a challenging topic. This stems, firstly, from the fact that the clustering algorithms can be useful for some graph types and not for others. Secondly, they are almost all computationally expensive because they need to re-compute several measures in each step (e.g. the betweenness for each edge). Nevertheless, nowadays, the real world applications need to deal with very large-scale graphs such as social networks or web graphs where the size grows exponentially (billions of vertices). For sake of scalability, some works try to provide a faster clustering solution by considering only local quantities, such as [61] but this is still not sufficient for real world applications. Yet, a major part of graph clustering algorithms seems to discard the evolving aspect of the real graphs. Indeed, the structure of graph in almost all domains changes over time by adding/removing edges and/or vertices.

3 Distributed Graph Processing Framework

3.1 Distributed Computation Framework

Parallel and distributed computing have been strongly studied during last twenty years and they have been tremendously popularized by new frameworks such as MapReduce [13,25,45,47,77] (see Section 3.2) and Dryad [34,60]. The notion of parallelism represents the ability to run simultaneously software in different processors in order to increase its performance while the distributed concept emphasizes the notion of loose coupling between those processors. The distributed

architectures can be described and classified according to the resources that the machines or the processors share with each other. This classification is particularly important if we speak about large graph storage and processing.

The main categories are shared-memory, shared-disk and share-nothing. The shared-memory architecture describes distributed systems that share a common memory space. In the case of distributed machines, it can be a distributed cache where the data is commonly available. In large-scale super-computer the processors can share the data through non-uniform Memory Access (NUMA) [36]. However, this kind of architectures is more suited for small parallel data problem, as the shared memory must manage the data consistency and accesses through the different clients. The second category is the shared-disk that enables to connect distributed processors by Local Area Network (LAN). Even if they are less costly in term of scalability when adding new storage nodes than the shared-memory, they still suffer from the access contention and the data consistency when number of processor clients dramatically increases. Finally the last category is the shared-nothing architecture in which each machine has its own independent storage. Therefore, a new important concept has been defined, *the partitioning*. The data are partitioned over the cluster of machines according to a partitioning policy. This policy defines the location of the data and then, the distributed computing framework can send dedicated tasks where the data is located. This represents the notion of data *locality*.

The parallel programming paradigm can be described as either explicit or implicit parallel programming. In the explicit version, the developer will have to explicitly create tasks, synchronization points, managing threads and processes and ensuring that the parallel operations will be safe, etc. On the other hand, in the implicit parallel programming, the developer does not need to care about these details. The compiler or the distributed framework handles all aspects related to the parallel execution such as the portion of the code that must run in parallel, the task location routing, the creation of threads and processes, the data access, etc. Although the explicit parallel programming is richer and let the developer accurately drive the distributed processing, it represents a serious complexity in term of design and implementation, and is error-prone. Most of the distributed processing frameworks presented in this paper are shared-nothing and expose an implicit programming model.

3.2 Large Graph Processing

The MapReduce technique, proposed by Google, is a famous paradigm in distributed computing on large data sets and can be used for computer programs that need to process and generate large amounts of data. For instance, MapReduce has been used by Google to create the index of all the crawled web pages. Hadoop [10] is an open-source implementation of MapReduce. In addition to the distributed computing, the three main strengths of Hadoop result in data locality, fault tolerance and parallel processing. For the sake of completeness, we define briefly the two major steps of the MapReduce paradigm:

- **Map**: in this step, the problem is partitioned into a set of small sub-problems. This set is then distributed over the machines available in the cluster and each sub-problem is processed, independently, in a single machine, namely worker node.
- **Reduce**: in this step, all the answers to all sub-problems are gathered from the worker nodes and are then merged to form the output solution.

These two functions are written by the user and are applied to distributed data over a cluster of machines as shown in Figure 1(a). This programming model offers to developers an easy and simple way to deal with large data sets in a distributed computation environment. Indeed, by means of Hadoop, the developers do not need to be experts on distributed computation, and they have just to focus on the design of their algorithms with MapReduce programming paradigm. Nevertheless, this paradigm is not well suited for graph processing tasks and iterative algorithms. In fact, a simple iterative algorithm needs a whole execution of a MapReduce task in each iteration which forms a huge data migration and computation over the algorithm execution, requiring lots of I/Os and unnecessary computations. Figure 1(b) illustrates two iterations with a naive implementation with MapReduce paradigm (in each iteration the data are partitioned, processed and the intermediate results are stored).

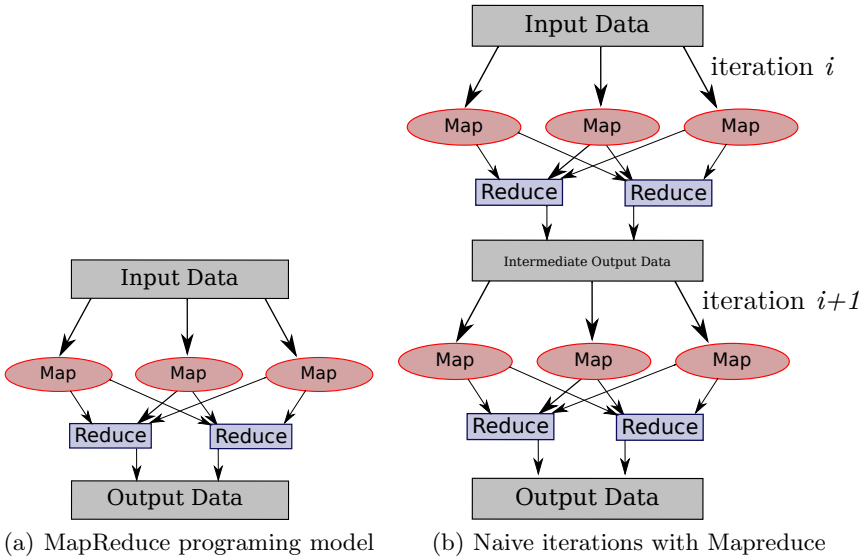


Fig. 1. MapReduce paradigm and a naive iteration implementation

In order to overcome this problem, some works [15,18,23,37,38,51,76] proposed a set of techniques to improve classical MapReduce for iterative algorithms. For instance, Twister [23] and Haloop [15] solutions reuse workers across iterations

by changing only the input which minimizes the number of instantiated workers. In addition, Hadoop supports caching of input and output of iterations to save I/Os and it uses a loop-aware scheduling of jobs which is a very interesting extension of Hadoop. Despite the improvements, these solutions lack efficiency for the graph-based algorithms since they deal essentially with multidimensional data. In this respect, some methods have been developed to deal especially with the distributed computation of linkage structural data [18,51]. In this context, the most popular framework was introduced by Google and called Pregel [51]. The main purpose of Pregel is to provide a distributed computation framework entirely dedicated to graph processing algorithms implementation. Google Pregel was inspired from the Bulk Synchronous Parallel (BSP) programming paradigm [71]. Roughly, in the BSP model an algorithm is executed as a sequence of *supersteps* separated by a global synchronization points until termination. Within each superstep a processor (or a virtual processor) may perform the following operations; (1) perform computations on a set of local data (only) and (2) send or receive messages. Similarly, in Pregel, within a *superstep* the vertices of graph execute the same user-defined function, in parallel. This function can include : a modification of the state of a vertex or that of its outgoing edges, read messages sent to the vertex in the previous superstep, send messages to other vertices that will be received in the next superstep, or even a modification of the topology of the graph (deleting or adding vertices and/or edges) [51]. Pregel uses a “*vertex voting to halt*” technique to determine the algorithm termination. Each vertex has two possible states: *active* or *inactive*. An algorithm is considered terminated when all the vertices are in the *inactive* state. Practically, in the initial superstep (Superstep 0), all vertices are in the *active* state, then in each subsequent superstep each vertex can vote to halt and then, explicitly deactivate itself. An *inactive* vertex do not participate on any superstep unless it receives an non-empty message⁴.

There exists several open-source implementations of Pregel, but the two mature ones are Apache Hama and Apache Giraph. In the remaining of section, we will address the implementation of two graph algorithms with Pregel paradigm: single source shortest path (SSSP) and pageRank.

SSSP Implementation: Algorithm 1 shows a pseudo-code of the vertex function for a SSSP implementation within Pregel framework. Initially, each vertex value (except the source), which corresponds to the distance to reach it from the source, is initialized to an infinity constant (larger value than any possible distance in the graph). In this algorithm, in each superstep, each vertex reads messages from its neighbors. Each message contains the distance between the source vertex and the current vertex (through a given adjacent vertex). For a given vertex, if the minimum message value is less than the actual associated value, the vertex updates its current value. Then, the vertex sends messages through all its outgoing edges, such that each message contains the sum of the weight of its outgoing edge and the new value associated to the vertex. Finally,

⁴ The fact of receiving a message activates the vertex state.

```

Input : Messages: Set of received messages
if currentVertex is the source then
  | minimumDistance  $\leftarrow$  0;
else
  | minimumDistance  $\leftarrow$   $\infty$ ;
end
foreach message  $m \in$  Messages do
  | minimumDistance  $\leftarrow$  minimum(minimumDistance, valueOf( $m$ ));
end
if minimumDistance  $\leq$  valueOf(currentVertex) then
  | valueOf(currentVertex)  $\leftarrow$  minimumDistance;
  | foreach outgoing edge  $e$  from currentVertex do
  | | sendMessageTo(targetOf( $e$ ), (minimumDistance+ valueOf( $e$ )));
  | end
end
VoteTohalt();

```

Algorithm 1. Pregel: Vertex function for Single source shortest path problem.

the vertex vote to halt. The algorithm terminates if there is no more updates performed. As result of the algorithm, each vertex of the graph will be associated to a value that denotes its minimum distance from the source vertex to it. In the case of unreachable vertex from the source (unconnected graph), the associated value to such vertex is set to the infinity constant.

For sake of more demystification, let us analyze an example of the execution of the previous algorithm. For this purpose, Figure 2 provides a superstep by superstep execution of the SSSP algorithm on a sample graph. Here, we consider the vertex labeled by (1) as the source. The initial step consists on setting the values associated to all the other vertices to infinity. In Superstep 1, the vertices (2), (3) and (4) receive from the vertex (1) (in Superstep 0), respectively, the messages containing their distances to (1). For instance, the vertex (2) receives a message that contains 6 which is the sum of the value of vertex (1) and the weight of outgoing edge ((1) \rightarrow (2)). Moreover, in Superstep 1, the source vertex is in inactive state because it does not receive any message in this superstep. The next supersteps follow the same procedure until all the vertices are in inactive state.

PageRank Implementation: Algorithm 2 shows a pseudo-code of the vertex function for a *PageRank* implementation within Pregel framework. Initially, each vertex value, which correspond to the *PageRank* estimation, is initialized to $\frac{1}{SizeOfGraph}$. In this algorithm, in each superstep, each vertex read messages from its neighbors. Each message contains tentative pageRank divided by the number of outgoing edges of the involved vertex. For a given vertex, the received message values are summed up into *sum*, then the vertex updates its current *PageRank* value by $\frac{0.15}{SizeOfGraph} + 0.85 \times sum$ (which follows Equation 2.1). Then, the vertex sends messages through all its outgoing edges. Finally, after a

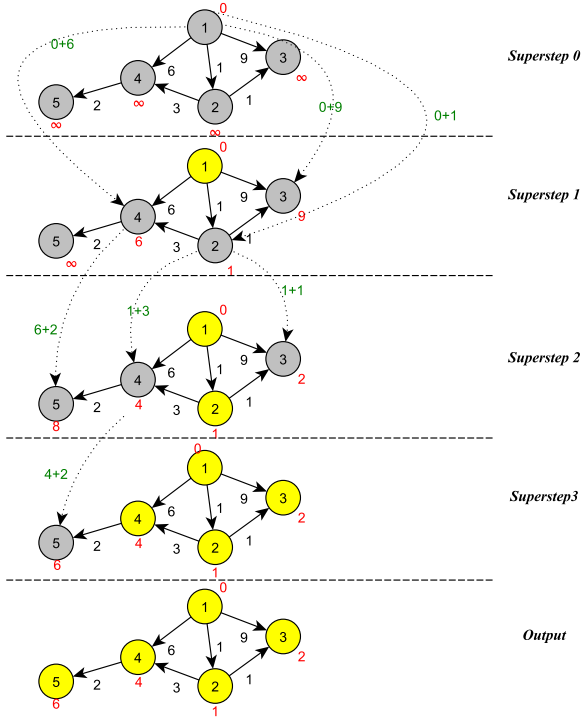


Fig. 2. Single source shortest path supersteps. Dotted lines describe how the messages are sent (from/to), the green labels are the message values. Yellow vertices have voted to halt.

fixed number of supersteps (iterations) the vertex vote to halt. The algorithm terminates if there is no more updates performed. As result of the algorithm, each vertex of the graph will be associated to a value that denotes its *PageRank*. As mentioned in [51], instead of fixing the number of iteration, one could find a suitable setup of this algorithm to run until convergence of *PageRank* values.

4 Graph Data Warehouses: An Emerging Challenge

Since we can provide efficient methods for processing and mining large graph, the next question to answer is how to store many of these large graphs and still exploiting their informational potential. Nowadays, we end up with a significant number of graphs in data warehouse, not because they are the easiest way to analyze the data but because they are the most meaningful way to represent the relationship concepts.

The concept of graph data warehouse is similar to the traditional data warehouse: the warehouse is fed by a set of data sources that can be either databases or existing graphs. Those data sources are extracted and transformed to be

```

Input : Messages: Set of received messages
if NumberOfSuperstep  $\geq 1$  then
  sum  $\leftarrow 0$ ;
  foreach message  $m \in$  Messages do
    | sum  $\leftarrow$  sum + valueOf( $m$ );
  end
  valueOf(currentVertex)  $\leftarrow 0.15 / \text{SizeOfGraph} + 0.85 \times$  sum;
end
if NumberOfSuperstep  $<$  MaximumNumberOfIteration then
  | N = SizeOf({outgoing edges from currentVertex})
  | sendMessageToAllNeighbors(valueOf(currentVertex) / N);
else
  | VoteTohalt();
end

```

Algorithm 2. Pregel: Vertex function for *PageRank*.

loaded as a graph structure in order to facilitate the data mining. The problem is then (1) how to merge different graphs from different data sources, (2) how to define an efficient conceptual modeling on top of graph data and finally, (3) how to express efficient queries. Let us take the example of a telecommunication operator who owns (1) a network address book service, (2) a chat service, (3) a social network, and (4) a set of call data records (CDRs). The objective of this telecommunication operator is to merge that information within on single aggregated graph in order to extract and infer information such as: influencers and maven, potential churners and service usage pattern. Therefore the first question is how to merge the different graphs from the address book, the chat service contacts, the social network and the CDRs, knowing that there are different services launched by different departments and using different ID for users. The second question would be how to define a conceptual model enabling to define roles in edges we can navigate, additivity for the time spent on each service, the navigation path on edge for clustering algorithm that can be used for maven identification, etc. Finally, we have the queries such as once the maven have been identified, retrieve the closest users based on their location or retrieve the average time spent on all services by influencers or potential churners. We can also consider composite queries such as (1) extracting all male influencers living in cities of more than 1M inhabitants, (2) extracting potential churners who are in the neighborhood of those influencers and (3) extract last year churning in the graph neighborhood of the current potential churning and evaluate their interaction through the social network service and CDR, (4) according to the query (2) and (3) defining the best influencers to propagate a promotional message.

This example leads to the key question of this section: if the graph modeling enables to better leverage the relationship concept in mining algorithms and if several execution engines and distributed frameworks enable to apply those mining algorithms on significant graphs, what is still missing for a graph data

warehouse? This section will answer to this question by first introducing basic foundation of data warehouse approach, from there we will explain the existing lack of methodologies, modeling approaches and analytic tools for an equally efficient graph data warehouse.

4.1 Using Relational Data Warehouse for Storing Graphs?

The data mining algorithms are involved in each step of the traditional data warehouse [33], we can use techniques for identifying key attributes or finding related measures or dimensions, some other techniques enable to limit the scope of the data extraction on specific clusters. Usually the OLAP framework is integrated with a mining framework in order to use both in conjunction, this integration is often named On-Line Analytic Mining (OLAM) and exploratory multi-dimensional data mining [31]. According to [31], there is at least four ways to use OLAM and OLAP in conjunction:

- using the multi-dimensional cube space for defining the data space for mining
- using OLAP queries for generating features and targets for mining
- using data mining as building blocs in a multi-step mining process
- using data cube computation for speeding-up repeated models construction

The graph model should be another way to represent the information and then it should be stored as any other data sources in the data warehouse. However, the graph model must be considered as a constraint. Indeed, if it is represented as a graph it is because it leverages the relationship concept. Let us take the example of a social network. We could represent it as traditional relational tables on which we can represent the concept of user as a table pointing to a m-n table defining all the relationship a user has. If we consider that, in average, in a social network a user has 100 friends, a simple request retrieving the friends of the friends will lead to 100^2 join requests. A social network can be stored without any issues on a relational data warehouse, but when comes the question about how we can leverage the relationship concept through a set of mining algorithms, we come to the conclusion that the graph model remains the most appropriate format. In addition, most of the mining techniques must navigate through the edge relations. As shown in Figure 3, this means that it (1) will significantly cost in terms of join operations and (2) will almost transfer the totality of the graph between the relational storage and the mining application, which could potentially represents a huge amount of data in the case of the graphs we consider in this paper.

We saw in the previous section how we can use new emerging distributed frameworks in order to apply traditional mining algorithms on significant graphs. This means that graph mining analysis such as classification, link-based analysis, object or link-based recommendation, trust computation can be applied on graph models. An interesting advantage of those frameworks is that they can both leverage the data locality, e.g., the information about the location of sharded data, and they can integrate mining algorithms within the distributed storage. As a result they highly minimize the quantity of data exchanged between the processing and the storage nodes during a mining operation.

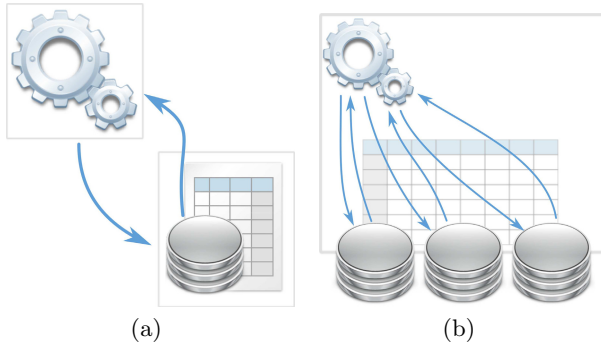


Fig. 3. (a) The traditional relational DB approach involves that almost all the graph content would be transferred between the mining application and the storage. (b) While in a data application server concept the mining application is implemented as an application in the distributed storage middleware.

4.2 Traditional Data Warehouse Approaches

For many years the knowledge presents in the data accumulated by an enterprise has always represented a key strategic element in its management, in term of business KPI, behavioral analysis, strategic marketing and many other fields. The traditional data warehouse aims at providing the software, the modeling approaches and the tool to analyze the set of data present in a collection a databases.

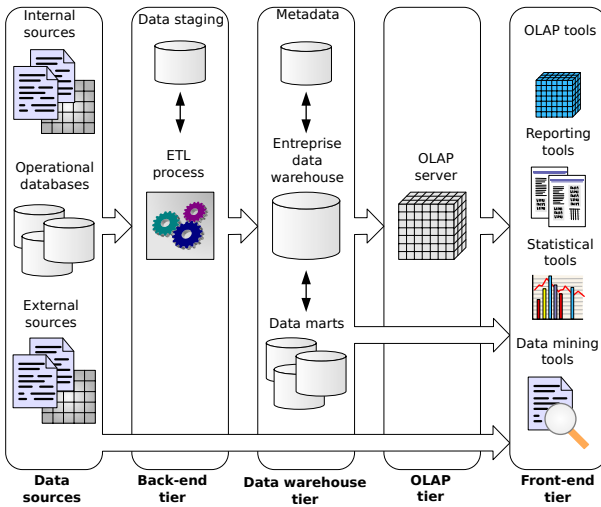


Fig. 4. A traditional process overview [52]

Figure 4 shows the traditional process to extract knowledge from data. A first step consists in extracting data from the collection of data sources that can be relational, files, remote web sources, etc. The ETL (Extract, Transform and Load) aims at structuring the extracted data in a more processable form, this is the data warehouse tier. This phase is usually done manually by using an ETL software, without specific modeling approach. However new emerging researches try to adopt a complete modeling process aligned with the modeling techniques used in the OLAP tier [3].

The data warehouse tier is then used for designing high level models that will be used for executing specific requests. This is traditionally the area of On-Line Analytical Processing (OLAP). The idea is to design a logical model suited for regrouping the data that are needed for the OLAP queries and to generate a related physical model. The OLAP queries take advantage of the physical model and the query model to generate a physical execution plan. The development of conceptual modeling of data warehouse has been an important research area that is still highly active. Most of the research topics focus on the improvement of the *snowflake* and *star* schema [53]. Some researches try to add a graphical representation [63] based on the ER model [65,70] or based on UML [149], other focus on models that enable to define different level of hierarchies [8,32,40,63], while others provide models that take into account the role played by a measure in different dimensions [41,49]. The model described in [53] tries to summarize the main limitations of the snowflake and star model and proposes a new model that includes most of the previous researches in this area.

The last phase of the processing is the OLAP tier in which the data processing is led by the expression of the OLAP queries. An overview of OLAP techniques has been described in [16]. The authors describe the main OLAP operations as roll-up (increasing the level of aggregation) and drill-down (decreasing the level of aggregation or increasing detail) along one or more dimension hierarchies, slice and dice (selection and projection), and pivot (re-orienting the multidimensional view of data). Usually the OLAP queries are expressed using standard SQL or Multidimensional Expression language (MDX) from Microsoft [58].

In conclusion we can summarize the complete data warehouse process by (1) the storage and (2) the process to extract, (3) to model and (4) to query the data. In the next section we will show that graph data warehouse is far from the maturity reached by the legacy warehouses.

4.3 Challenges in Graph Data Warehouses

In previous sections we have shown that distributed processing frameworks can be used to implement graph mining algorithms. However, this area lacks dramatically an unified conceptual approach to mine graph as found in legacy data warehouses. Figure 5 shows an equally functional data warehouse process when dealing with graphs. Similarly the data sources can be existing databases, as it is the case in fraud detection or call data records in telecommunication or even existing graphs. After an ETL process, we end-up with a consolidated graph that is the integration of different graphs. This graph must be queried and

analyzed. Currently, there is no common approach that enables to model an equally functional conceptual modeling approach such as the multi-dimensional cube for graphs. We need to be able to model an intermediate structure keeping the relationship as a central concept but enabling to represent different navigation paths, different roles in those paths while being able to represent hierarchies. There is a real gap in the research community in this area although the need for this kind of conceptual modeling approach will grow in the coming years. However, existing conceptual modeling approaches such as the Multidimensional model [53] should be independent from the underlying physical model, but it will need to be extended for specific graph semantics such as navigation paths, role in relationships and other properties such as the additivity in the navigation path. In the same way, there is no graph query language giving the same level of flexibility as the OLAP query language. Although few graph languages exist, they have been designed with specific objectives in mind and do not really represent an equally functional OLAP query language. We can cite Gremlin [4] that comes from the graph database area and SparQL [30,67] which defines standard query language and data access protocol, mainly used for RDF meta-models. Finally, the graph mining and data warehouses need to have an integrated execution processing framework. As in legacy OLAP, we need to generate, from the graph query, a logical execution plan and finally, a physical execution plan taking advantage of the distributed aspects of the graph. Currently, there is no way to express a graph query and to generate a distributed execution plan as it can be found in Apache PigLatin. However, few works especially in the web semantic field try to leverage existing distributed processing frameworks such as MapReduce [66], but it is highly dependent to RDF and does not really leverage the graph aspects. As for the conceptual modeling, this is clearly a gap in the research community.

It is worth noting that, at the moment of writing this paper and at our level of knowledge, the most complete research project on graph data warehouse is GraphCube [74]. The authors defined the concept of multidimensional network which is a graph on which each vertex is a tuple in a table. The attributes of

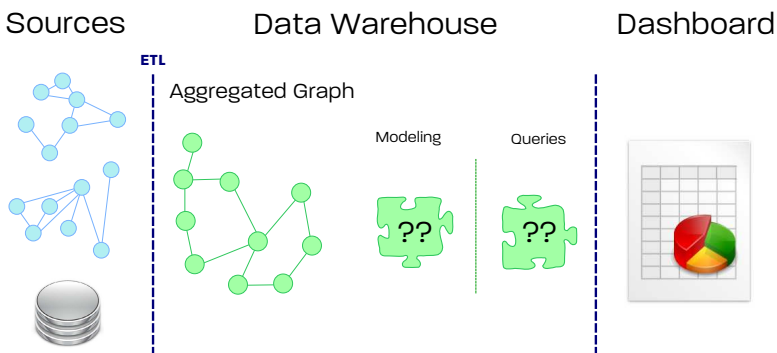


Fig. 5. The graph data warehouse process

this table represent the multidimensional space. The authors showed that we can execute the same OLAP queries on a table and a corresponding graph. This work enables to mine in the same time traditional relational sources and a graph as an additional source of information. They also defined the algorithms to obtain the different aggregated networks from queries. Finally they present the materialization approach. Although this paper is the most advanced research in graph data warehouse, there is still open questions. Indeed, (1) they only consider a graph of vertex of the same type that let all the open questions we introduced unanswered. (2) They consider only local centralized processing and (3) the materialization policy is inspired by legacy data warehouse and does not leverage the distributed graph processing aspects.

5 Conclusion

The large graph mining is becoming an important requirement coming from the industry and the research community. The graph model leverages the relationship between objects and enables to better structure linked data. In the other hand this model is not well suited for traditional data mining algorithms and processing framework. The mining algorithms need to be re-designed to take into account the structural nature of graphs and to be adapted to distributed programming paradigms to scale. In this paper we presented the *PageRank*, the k-means algorithm and the centrality measure, we described them and explained how they can be adapted for graph structures. Afterwards, we introduced new emerging graph distributed frameworks and described how the previous mining algorithms can be implemented within their programming models. Finally, we introduced a new field of research, the graph data warehouse, which is deeply linked with large graph mining. This field is still at early stage but dramatically lacks conceptual modeling, unified query model and integration with new distributed programming techniques.

This paper shown that if we consider to analyze significant graphs, distributed programming techniques can be applied to efficiently speed-up the processing. However, the implementation of those techniques and the frameworks is still at early stage, the documentation and the support are clearly lacking. In addition, the implicit distributed programming model can be difficult for porting legacy graph mining algorithms.

References

1. Abelló, A., Samos, J., Saltor, F.: YAM²: a multidimensional conceptual model extending UML. *Inf. Syst.* 31(6), 541–567 (2006)
2. Aggarwal, C.C., Wang, H. (eds.): *Managing and Mining Graph Data*. *Advances in Database Systems*, vol. 40. Springer (2010)
3. Akkaoui, Z.E., Zimányi, E., Mazón, J.-N., Trujillo, J.: A model-driven framework for ETL process development. In: *Proceedings of the 14th ACM International Workshop on Data Warehousing and OLAP, DOLAP 2011*, pp. 45–52. ACM (2011)

4. Avram, A.: Gremlin, a language for working with graphs. Technical report, InfoQ (2010), <http://www.infoq.com/news/2010/01/Gremlin>
5. Bader, D.: Analyzing Massive Social Networks Using Multicore and Multithreaded Architectures. In: Keller, R., Kramer, D., Weiss, J.-P. (eds.) Facing the Multicore-Challenge. LNCS, vol. 6310, p. 1. Springer, Heidelberg (2010)
6. Bader, D.A., Madduri, K.: Snap, small-world network analysis and partitioning: An open-source parallel graph framework for the exploration of large-scale networks. In: Proceedings of the IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2008, pp. 1–12. IEEE (2008)
7. Balakrishnan, A., Magnanti, T.L., Wong, R.T.: A Dual-Ascent procedure for Large-Scale uncapacitated network design. *Operations Research* 37(5), 716–740 (1989)
8. Bauer, A., Hümmer, W., Lehner, W.: An Alternative Relational OLAP Modeling Approach. In: Kambayashi, Y., Mohania, M., Tjoa, A.M. (eds.) DaWaK 2000. LNCS, vol. 1874, pp. 189–198. Springer, Heidelberg (2000)
9. Bellman, R.: On a routing problem. *Quarterly of Applied Mathematics* 16, 87–90 (1958)
10. Bialecki, A., Cafarella, M., Cutting, D., O’Malley, O.: Hadoop: A framework for running applications on large clusters built of commodity hardware (2005), <http://lucene.apache.org/hadoop/>
11. Borodin, A., Roberts, G.O., Rosenthal, J.S., Tsaparas, P.: Finding authorities and hubs from link structures on the World Wide Web. In: Proceedings of the 10th International Conference on World Wide Web, WWW 2001, pp. 415–429. ACM (2001)
12. Botafogo, R.A., Rivlin, E., Shneiderman, B.: Structural analysis of hypertexts: Identifying hierarchies and useful metrics. *ACM Trans. Inf. Syst.* 10(2), 142–180 (1992)
13. Brants, T., Popat, A.C., Xu, P., Och, F.J., Dean, J.: Large language models in machine translation. In: Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, EMNLP-CoNLL 2007, pp. 858–867. ACL (2007)
14. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. *Computer Networks* 30(1-7), 107–117 (1998)
15. Bu, Y., Howe, B., Balazinska, M., Ernst, M.D.: Haloop: efficient iterative data processing on large clusters. *Proceedings of the VLDB Endowment* 3, 285–296 (2010)
16. Chaudhuri, S., Dayal, U.: An overview of data warehousing and OLAP technology. *SIGMOD Record* 26(1), 65–74 (1997)
17. Chen, M.-S., Han, J., Yu, P.S.: Data mining: An overview from a database perspective. *IEEE Trans. Knowl. Data Eng.* 8(6), 866–883 (1996)
18. Chen, R., Weng, X., He, B., Yang, M.: Large graph processing in the cloud. In: Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010, pp. 1123–1126. ACM (2010)
19. Chung, F.R.K.: A local graph partitioning algorithm using heat kernel pagerank. *Internet Mathematics* 6(3), 315–330 (2009)
20. Cohn, D., Chang, H.: Learning to probabilistically identify authoritative documents. In: Proceedings of the Twenty-Fourth International Conference on Machine Learning, ICML 2007, pp. 167–174. Morgan Kaufmann (2007)
21. Datta, D., Figueira, J.R.: Graph partitioning by multi-objective real-valued metaheuristics: A comparative study. *Appl. Soft Comput.* 11(5), 3976–3987 (2011)
22. Dijkstra, E.W.: A note on two problems in connexion with graphs. *Numerische Mathematik* 1(1), 269–271 (1959)

23. Ekanayake, J., Li, H., Zhang, B., Gunarathne, T., Bae, S.-H., Qiu, J., Fox, G.: Twister: a runtime for iterative mapreduce. In: Proceedings of the 19th ACM International Symposium on High Performance Distributed Computing, HPDC 2010, pp. 810–818. ACM (2010)
24. Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press (1996)
25. Fedak, G., Fox, G., Antoniu, G., He, H.: Future of mapreduce for scientific computing. In: Proceedings of the Second International Workshop on MapReduce and its Applications, MapReduce 2011, pp. 75–76. ACM (2011)
26. Floyd, R.W.: Algorithm 97: Shortest path. *Commun. ACM* 5, 345 (1962)
27. Freeman, L.: Centrality in social networks conceptual clarification. *Social Networks* 1(3), 215–239 (1979)
28. Gaujal, B., Navet, N., Walsh, C.: Shortest-path algorithms for real-time scheduling of FIFO tasks with minimal energy use. *ACM Trans. Embed. Comput. Syst.* 4, 907–933 (2005)
29. Girvan, M., Newman, M.E.J.: Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99(12), 7821–7826 (2002)
30. Gupta, R., Malik, S.K.: SPARQL semantics and execution analysis in semantic web using various tools. In: Proceedings of the 2011 International Conference on Communication Systems and Network Technologies, CSNT 2011, pp. 278–282. IEEE Computer Society (2011)
31. Han, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Morgan Kaufmann (2000)
32. Husemann, B., Lechtenböcker, J., Vossen, G.: Conceptual data warehouse design. In: Proceedings of the International Workshop on Design and Management of Data Warehouses, DMDW 2000, pp. 3–9 (2000)
33. Imielinski, T., Mannila, H.: A database perspective on knowledge discovery. *Commun. ACM* 39(11), 58–64 (1996)
34. Isard, M., Budi, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: distributed data-parallel programs from sequential building blocks. In: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, EuroSys 2007, pp. 59–72. ACM (2007)
35. Jain, A., Murty, M., Flynn, P.: Data clustering: a review. *ACM Computing Surveys (CSUR)* 31(3), 264–323 (1999)
36. Kaeli, D.R., Fong, L.L., Booth, R.C., Imming, K.C., Weigel, J.P.: Performance analysis on a cc-numa prototype. *IBM J. Res. Dev.* 41, 205–214 (1997)
37. Kang, U., Tsourakakis, C., Appel, A., Faloutsos, C., Leskovec, J.: Hadi: Fast diameter estimation and mining in massive graphs with hadoop. *CMU-ML-08-117* (2008)
38. Kang, U., Tsourakakis, C.E., Faloutsos, C.: PEGASUS: A peta-scale graph mining system. In: Proceedings of the Ninth IEEE International Conference on Data Mining, ICDM 2009, pp. 229–238. IEEE Computer Society (2009)
39. Kaufman, L., Rousseeuw, P.J.: *Finding Groups in Data: An Introduction to Cluster Analysis*. Wiley-Interscience (2005)
40. Khosrow-Pour, M. (ed.): *Encyclopedia of Information Science and Technology*, 5 volumes. Idea Group (2005)
41. Kimball, R., Ross, M.: *The Data Warehouse Toolkit: The Complete Guide to Dimensional Modeling*, 2nd edn. John Wiley & Sons, Inc. (2002)
42. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. In: Proceedings of the Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 1998, pp. 668–677. ACM/SIAM (1998)

43. Kleinberg, J.M.: Authoritative sources in a hyperlinked environment. *J. ACM* 46, 604–632 (1999)
44. Kumar, R., Novak, J., Tomkins, A.: Structure and evolution of online social networks. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2006*, pp. 611–617. ACM (2006)
45. Lämmel, R.: Google’s mapreduce programming model revisited. *Sci. Comput. Program.* 70, 1–30 (2008)
46. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Statistical properties of community structure in large social and information networks. In: *Proceedings of the 17th International Conference on World Wide Web, WWW 2008*, pp. 695–704. ACM (2008)
47. Liu, C., Guo, F., Faloutsos, C.: Bbm: bayesian browsing model from petabyte-scale data. In: *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2009*, pp. 537–546. ACM (2009)
48. Lorenz, D.H., Orda, A.: Qos routing in networks with uncertain parameters. *IEEE/ACM Trans. Netw.* 6, 768–778 (1998)
49. Luján-Mora, S., Trujillo, J., Song, I.-Y.: A uml profile for multidimensional modeling in data warehouses. *Data Knowl. Eng.* 59(3), 725–769 (2006)
50. MacQueen, J.B.: Some methods for classification and analysis of multivariate observations. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, pp. 281–297. University of California Press (1967)
51. Malewicz, G., Austern, M.H., Bik, A.J.C., Dehnert, J.C., Horn, I., Leiser, N., Czajkowski, G.: Pregel: a system for large-scale graph processing. In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2010*, pp. 135–146. ACM (2010)
52. Malinowski, E., Zimányi, E.: *Advanced data warehouse design: From conventional to spatial and temporal applications*. Springer (2008)
53. Malinowski, E., Zimányi, E.: Multidimensional conceptual modeling. In: Wang, J. (ed.) *Encyclopedia of Data Warehousing and Mining*, 2nd edn., pp. 293–300. IGI Global (2008)
54. Marchiori, M.: The quest for correct information on the web: Hyper search engines. *Computer Networks* 29(8-13), 1225–1236 (1997)
55. Martínez-Bazan, N., Muntés-Mulero, V., Gómez-Villamor, S., Nin, J., Sánchez-Martínez, M.-A., Larriba-Pey, J.-L.: Dex: high-performance exploration on large graphs for information retrieval. In: *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007*, pp. 573–582. ACM (2007)
56. McSherry, F.: Spectral partitioning of random graphs. In: *Proceedings of the 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, pp. 529–537 (2001)
57. Mortensen, E.N., Barrett, W.A.: Interactive segmentation with intelligent scissors. *Graph. Models Image Process.* 60, 349–384 (1998)
58. Nolan, C.: *Manipulate and query OLAP data using ADOMD and multidimensional expressions*. Technical report, Microsoft Research (1999)
59. Page, L., Brin, S., Motwani, R., Winograd, T.: The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab. Previous number = SIDL-WP-1999-0120 (November 1999)
60. Qiu, X., Ekanayake, J., Beason, S., Gunarathne, T., Fox, G., Barga, R., Gannon, D.: Cloud technologies for bioinformatics applications. In: *Proceedings of the 2nd Workshop on Many-Task Computing on Grids and Supercomputers, MTAGS 2009*, pp. 6:1–6:10. ACM (2009)

61. Radicchi, F., Castellano, C., Cecconi, F., Loreto, V., Parisi, D.: Defining and identifying communities in networks. *Proceedings of the National Academy of Sciences of the United States of America* 101(9), 2658 (2004)
62. Rattigan, M.J., Maier, M.E., Jensen, D.: Graph clustering with network structure indices. In: *Proceedings of the Twenty-Fourth International Conference on Machine Learning, ICML 2007*, pp. 783–790. ACM (2007)
63. Rizzi, S.: Conceptual modeling solutions for the data warehouse. In: Erickson, J. (ed.) *Database Technologies: Concepts, Methodologies, Tools, and Applications*, pp. 86–104. IGI Global (2009)
64. Rodriguez, M.A., Neubauer, P.: A path algebra for multi-relational graphs. In: *Proceedings of the 2011 IEEE 27th International Conference on Data Engineering Workshops, ICDEW 2011*, pp. 128–131. IEEE Computer Society (2011)
65. Sapia, C., Blaschka, M., Höfling, G., Dinter, B.: Extending the E/R Model for the Multidimensional Paradigm. In: Kambayashi, Y., Lee, D.-L., Lim, E.-P., Mohania, M., Masunaga, Y. (eds.) *ER 1998*. LNCS, vol. 1552, pp. 105–116. Springer, Heidelberg (1999)
66. Schätzle, A., Przyjaciół-Zablocki, M., Lausen, G.: PigSPARQL: mapping SPARQL to Pig Latin. In: *Proceedings of the International Workshop on Semantic Web Information Management, SWIM 2011*, pp. 4:1–4:8. ACM (2011)
67. Segaran, T., Evans, C., Taylor, J.: *Programming the Semantic Web - Build Flexible Applications with Graph Data*. O'Reilly (2009)
68. Sommer, C.: *Approximate Shortest Path and Distance Queries in Networks*. PhD thesis, University of Tokyo (2010)
69. Sui, X., Nguyen, D., Burtscher, M., Pingali, K.: Parallel Graph Partitioning on Multicore Architectures. In: Cooper, K., Mellor-Crummey, J., Sarkar, V. (eds.) *LCPC 2010*. LNCS, vol. 6548, pp. 246–260. Springer, Heidelberg (2011)
70. Tryfona, N., Busborg, F., Christiansen, J.G.B.: Starer: A conceptual model for data warehouse design. In: *Proceedings of the Second ACM International Workshop on Data Warehousing and OLAP, DOLAP 1999*, pp. 3–8. ACM (1999)
71. Valiant, L.G.: A bridging model for parallel computation. *Commun. ACM* 33, 103–111 (1990)
72. Wasserman, S., Faust, K.: *Social Network Analysis: Methods and Applications*. Structural analysis in the social sciences, vol. 8. Cambridge University Press (1994)
73. Zhan, F.B., Noon, C.E.: Shortest path algorithms: An evaluation using real road networks. *Transportation Science* 32, 65–73 (1998)
74. Zhao, P., Li, X., Xin, D., Han, J.: Graph cube: on warehousing and OLAP multidimensional networks. In: *Proceedings of the 2011 International Conference on Management of Data, SIGMOD 2011*, pp. 853–864. ACM (2011)
75. Zhao, X., Sala, A., Wilson, C., Zheng, H., Zhao, B.Y.: Orion: shortest path estimation for large social graphs. In: *Proceedings of the 3rd Conference on Online Social Networks, WOSN 2010*, p. 9. USENIX Association (2010)
76. Zhou, A., Qian, W., Tao, D., Ma, Q.: Disg: A distributed graph repository for web infrastructure (invited paper). In: *Proceedings of the Second International Symposium on Universal Communication, ISUC 2008*, pp. 141–145. IEEE Computer Society (2008)
77. Zhuang, L., Dunagan, J., Simon, D.R., Wang, H.J., Tygar, J.D.: Characterizing botnets from email spam records. In: *Proceedings of the 1st Usenix Workshop on Large-Scale Exploits and Emergent Threats*, pp. 2:1–2:9. USENIX Association (2008)

Big Data Analytics on Modern Hardware Architectures: A Technology Survey

Michael Saecker and Volker Markl

Technische Universität Berlin
Berlin, Germany

`firstname.lastname@tu-berlin.de`

Abstract. Big Data Analytics has the goal to analyze massive datasets, which increasingly occur in web-scale business intelligence problems. The common strategy to handle these workloads is to distribute the processing utilizing massive parallel analysis systems or to use big machines able to handle the workload. We discuss massively parallel analysis systems and their programming models. Furthermore, we discuss the application of modern hardware architectures for database processing. Today, many different hardware architectures apart from traditional CPUs can be used to process data. GPUs or FPGAs, among other new hardware, are usually employed as co-processors to accelerate query execution. The common point of these architectures is their massive inherent parallelism as well as a different programming model compared to the classical von Neumann CPUs. Such hardware architectures offer the processing capability to distribute the workload among the CPU and other processors, and enable systems to process bigger workloads.

Keywords: Modern Hardware Architectures, GPGPU, GPU, APU, FPGA, DBMS, Big Data Analytics.

1 Introduction

The term "Big Data" refers to the exceptional big size of data sets that are collected nowadays. Today, we observe an exponential growth of data sets, e.g., an institute like CERN produces 15 PB of data each year running their Large Hadron Collider [1].

Big Data Analytics describes the process of performing complex analytical tasks on such data which usually includes grouping, aggregation, or iterative processes. A report by McKinsey states that Big Data Analytics skills will ensure employability of computer science students and data analysts. In the United States alone, there is a need for 140,000 to 190,000 people with deep analytical skills [2].

The range of applications for Big Data Analytics is wide and due to space reasons, we will give only a few examples. The scenarios range from web logs, ETL processes [3], and general statistics [4] to data cleansing, machine learning [5,6],

or graph mining [7]. Other scenarios are sensor networks, spatial observations, triangle enumeration¹ [8], pairwise shortest path enumeration, or clustering [9].

The popularity of Big Data Analytic systems - which are often available as open-source - has not remained unnoticed by big companies. Google uses MapReduce - a parallel programming and execution model - for PageRank and inverted indexes. Facebook uses Hadoop - a popular open-source massively parallel analysis framework - to analyze their data and created Hive, a data warehouse system for Hadoop. eBay uses Hadoop for search optimization and Twitter uses Hadoop for log file analysis and other generated data [10,11]. This list is only a glimpse of the manifold real world appliances of such systems and the common factor among them is the huge amount of data they have to process.

In the future, the amount of data will increase even further with smart grids monitoring energy traffic [12], audio and video analysis, and smart houses.

This poses several problems for analysis software [13]:

- The amount of data is steadily increasing at a high speed, yet data should be up-to-date for analysis tasks.
- The response time of a query grows with the amount of data. At the same time, latencies must be reduced to provide actionable intelligence at the right time.
- Analysis tasks need to produce query results on large data sets in an adequate amount of time.

Generally, there are two approaches in tackling the problem of big data sets: horizontal scaling and vertical scaling. When scaling vertically, a server is equipped with faster hardware, more central processing units (CPUs), and/or more memory. Therefore, scaling up can be handled transparently² by the analysis software but requires substantial financial investments at one point in time. Furthermore, to cope with future workloads, the system needs to be adequately powerful, and initially, the additional performance goes to waste. Scaling horizontally, on the other hand, means distributing the work across many servers, often commodity machines. This increases performance in smaller steps and the financial investment to upgrade is by far smaller. The drawback of this approach is that the analysis software on top has to handle distribution by itself. Parallel programming adapts a divide and conquer approach: The problem is split into independently processable partitions (data parallelism). The resulting intermediate results need to be merged into a final result at the end, which requires synchronization between the parallel processors. Therefore, the speed-up of parallel programs is limited by the sequential parts, as stated by Amdahl's law [14] and shown in Figure 1.

A different kind of scaling is scaling in. It is the integration of multiple smaller sets of multiple tenants³ onto a single machine or process (multi-tenancy). Scal-

¹ It is used as a preprocessing step for methods to identify highly connected subgraphs or cliques in a graph.

² Assuming that the system is able to use multiple cores.

³ A tenant is a customer, which has its own view, data, and schema.

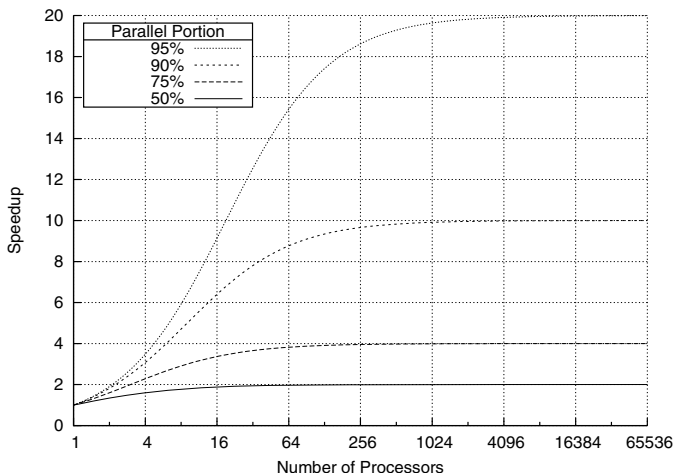


Fig. 1. Amdahl's law for varying number of cores and degrees of parallelism

ing in improves resource utilization, but introduces the problem of isolation between different tenants.

1.1 Scope of the Article

This article gives an overview of existing approaches to address the problems arising from Big Data Analytics when using modern hardware architectures. Therefore, a presentation of analysis systems is mandatory. Due to the large amount of existing systems and publications in this sector, we will focus on a representative sample of massively parallel analysis systems. We do not cover operational systems and their techniques like consensus protocols or distributed hash tables (DHTs) [15].

We will discuss different hardware architectures on an abstract level to give the reader insight to the possibilities and problems of the architectures. We restrict the scope of this article to processing elements and do not cover storage units, e.g., flash memory/solid-state drives (SSDs) or phase-change memory. We cannot give detailed, low-level explanations of all hardware components, and refer to the extensive references provided.

1.2 Outline

In Section 2, we will describe CPU architectures and discuss relational database systems (RDBMSs) along with problems in the context of Big Data Analytics. Then, we will describe the two state-of-the-art approaches to address these problems. In Section 3, we give an overview of massively parallel data analytic platforms that employ horizontal scaling. In Section 4, we explain modern hardware architectures in regard to vertical scaling. Finally, we will state our view of future development in this area and conclude in Section 5.

2 Background

This section gives a definition of relational database systems. We briefly describe the architecture of CPUs to explain the shortcomings of current architectures and how they impact DBMS performance.

2.1 CPU Architecture

Modern computer systems follow the von Neumann architecture and are designed to execute arbitrary programs. According to Moore's Law the number of transistors doubles every two years⁴.

Transistor growth was translated into higher frequencies of processors, but also used to implement techniques like out-of-order execution, branch prediction and instruction-level parallelism (ILP) to further increase the performance of CPUs. While CPUs evolved pretty fast, the improvement of random access memory (RAM) advanced only slowly. The gap between CPU clock speed and data transfer rate from RAM widened. To lighten the impact of slow RAM access, CPU developers included a cache hierarchy consisting of a small and very fast L1 cache up to a bigger and slower L3 cache. Figure 2(a) depicts the layout of the cache hierarchy for a single core.

A request for data is checked in the smallest cache level and then propagates through the levels. Therefore, a L2 cache miss automatically includes an L1 cache miss. If the data is not found in the last cache level - typically L3 cache nowadays - another cache needs to be accessed, the Translation Lookaside Buffer (TLB). The TLB stores address translations from virtual to physical addresses. If the requested data is on a page that is not contained in the TLB⁵, the address has to be calculated first. Hence, the worst case scenario is a TLB miss which can account for several hundred CPU cycles.

Today, it is very hard to increase the performance of single cores further because of the following problems:

- **Power wall:** To obtain higher clock frequencies, an increase in power would be necessary, but heat dissipation rises too immensely for practical solutions.
- **ILP wall:** Extracting instructions out of a single stream of instructions that can be issued in parallel gets harder and scales poorly.
- **Memory wall:** While the performance of CPUs steadily increases at a fast pace, the memory only gained minor performance over the years resulting in a gap between the processor processing speed and the speed to retrieve data from memory. This results in data-intensive applications to become bandwidth bound. Today, a high-end CPU is connected to RAM with a theoretical bandwidth of about 25 GB/s [18] which forms the upper bound of throughput for data-intensive applications.

⁴ In 1965, Moore initially stated that the number of transistors would double each year. He revised his statement to double each two years in 1975 [16].

⁵ Today, the TLB may also be realized as a multi-level cache, e.g., in Intel's Nehalem architecture [17].

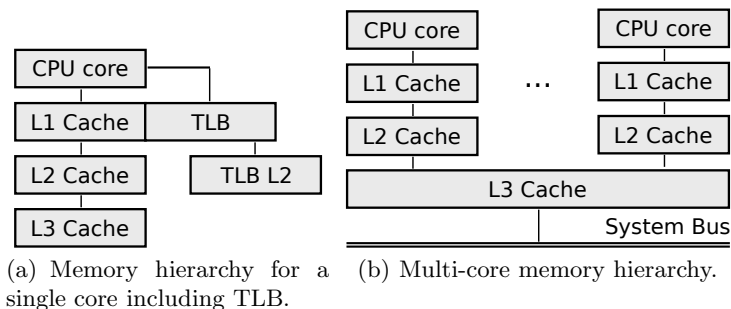


Fig. 2. Abstract layout of a modern CPU cache hierarchy

With single core performance stagnating, hardware developers shifted to multi-core CPUs to overcome this performance barrier. Recent generations of CPUs already come with a double digit number of physical cores per CPU (e.g., Intel E7-8870: 10 Cores [19], AMD Opteron 6282 SE: 16 Cores [20]). Typically, CPU cores have a private L1 and L2 cache, and share the L3 cache. Figure 2(b) shows the cache hierarchy for a multi-core CPU leaving out TLB caches for a better overview.

2.2 RDBMSs

In 1970, Edgar F. Codd published his idea of a relational model [21] which laid the foundation of today's database systems and was first realized in System R [22] and Ingres [23] in 1975/1976. Nowadays, relational databases are the de-facto standard for data management. Classical DBMSs use an n-ary storage model (NSM) to provide intra record locality. A relation consists of tuples with a defined set of attributes. In the NSM, tuples are stored consecutively with all their attributes. This performs nicely in a transactional system, yet for analysis only a fraction of the record data is needed. An alternative to NSM is the Decomposition Storage Model (DSM) [24], resulting in so called Column-Stores. In the DSM, tuples are split into their attributes. The values of an attribute for all tuples are stored consecutively. To reconstruct the original tuple, all attributes at a certain index need to be recombined. The benefit of storing and accessing data column-wise is that the bandwidth can be used more efficiently, compression ratio on columns is higher, and cache lines⁶ are used more efficiently. Figure 3 depicts the storage layout for both models.

With the development of ever faster hardware and increasing RAM sizes, the first in-memory databases were developed. Among them, MonetDB [25] pioneered in-memory column-store technology with cost models for cache and TLB misses. Boncz et al. [26] and Ailamaki et al. [27] identified cache and TLB misses to be the bottleneck of modern systems.

⁶ A cache line is a sequence of bytes transferred for a memory request that usually exceeds the amount of requested data to reduce the number of memory transactions for sequentially accessed data.

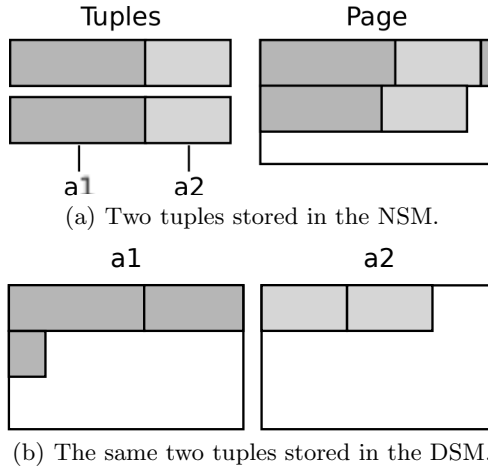


Fig. 3. Two tuples consisting of two attributes `a1` and `a2` are stored in NSM and DSM. In Figure 3(a) both tuples are stored consecutively in the page. In Figure 3(b) the tuples are split into their attributes and stored in sub-relations.

Recent developments include SAP Hana [28], C-Store [29], Vertica [30], and Vectorwise [31].

3 Horizontal Scaling

Horizontal scaling describes the process of distributing a task among many machines to speed up execution. In contrast to scaling vertically, the distribution of data and workload has to be programmed explicitly into the analysis software. Different approaches have evolved over the time: parallel systems using an SQL-like interface and massive data analysis systems which can be divided into categories of parallel data flow systems.

In this section, we will introduce these frameworks for data analysis tasks to give a broad overview of available systems. Table 1 gives an overview of the different components of the different software stacks.

3.1 Parallel DBMSs

GRACE, Gamma and Teradata are pioneers of parallel database systems. GRACE [32] is a parallel relational database system adopting data stream oriented processing. The system employs an architecture with two ring buses. The first ring handles filtering, projection and data distribution. The other ring performs processing.

Gamma [33] is a parallel relational database machine featuring a shared-nothing architecture of processor/disk pairs connected by a token ring. Gamma maximizes local data processing to achieve high I/O bandwidth and to reduce

Table 1. Overview of the different software stacks

| Component | Hadoop | Stratosphere | Dryad | Asterix |
|-----------------------|-----------------|--------------|------------------|---------|
| Higher-level Language | Pig, Hive, Jaql | Jaql | DryadLINQ, SCOPE | AQL |
| Programming Model | MapReduce | PACTs | - | - |
| Execution Engine | Hadoop MR | Nephele | Dryad | Hyracks |

the communication overhead. To permit local processing and query optimization, input data is partitioned among the processing/disk pairs, e.g., a selection is performed only at sites holding relevant data.

Teradata [34], the only commercial system of the three, started building a database management system for decision support using a parallel shared-nothing architecture and multiple processors. Their first prototype was shipped in 1983 and by 1996, a Teradata database was the world's largest database with eleven terabytes. Today, Teradata is one of the leading companies for data warehousing and analytical processing.

Other parallel DBMSs for Big Data Analytics include Greenplum, Aster Data and ParStream.

The Greenplum Unified Analytics Platform [35] includes the Greenplum database component for structured data, the Greenplum Hadoop integration for unstructured data, and the Greenplum Chorus productivity layer. The Greenplum Database was designed for Business Intelligence (BI) and analytical processing and utilizes a shared-nothing massively parallel processing architecture.

Aster Data [36,37] employs a massively parallel processing (MPP) architecture featuring a hybrid row and column store. Aster Data combines the MPP architecture with an implementation of a MapReduce processing framework resulting in a fault-tolerant, massively parallel environment for analysts.

ParStream [38] is a parallel database system that offers fast online analysis of structured data by incorporating GPUs for data processing. An important feature of ParStream is its innovative index structure which enables efficient parallel processing.

3.2 MapReduce

In 2004, Google published MapReduce, a programming and execution model to process large data sets of key/value pairs. MapReduce programs consist of two second-order functions, map and reduce, which originate from functional programming. Users provide first-order functions as arguments for map and reduce. Map passes each key/value pair to the first-order function separately to produce an intermediate result which is passed to the reduce function. Reduce passes all

key/value pairs sharing the same key to a single function call of its first-order argument function. Each function call emits zero, one or multiple key/value pairs. It is important to note, that the first-order functions need to be completely independent, i.e., functional without side effects, of each other as the paradigm depends on data parallelism.

The execution engine operates with a distributed file system - the Google File System (GFS) [39] - which handles data replication and distribution. A drawback of the execution engine is that both map and reduce take only one input, therefore operations like joins can only be realized using hacks or detours like concatenating two inputs and tagging them. The complete join logic must be hard-wired in a user-defined function, hence, depending on the input, the processing strategy may not be optimal. In contrast to RDBMS, there is no query optimizer that would choose join orders or physical strategies like hash join, merge join, or nested loop join. Also, the scheduling is fixed to a map step followed by a reduce step. This fixed scheduling and the limitation to two second-order functions removes most of the data-dependent optimization potential that could arise from reordering or combining operations.

Writing efficient programs requires in-depth knowledge of the programming model and the execution engine. Higher-level languages on top of MapReduce have been created to ease access to the parallel execution engine and improve processing efficiency. We will describe Pig, Hive and Jaql which build on Hadoop [40] - an open-source implementation of MapReduce - as three examples.

Pig. Formerly Yahoo! Research's, now Apache's Pig project [41,42] consists of the execution engine Pig and the data flow language PigLatin, which offers a declarative, SQL-like approach with imperative elements. Pig features a data model consisting of atoms, tuples, bags, and maps that allows for the nesting of data structures. Bags in Pig not only allow for duplicates as in common bags but also tuples in the bag do not have to share a structure or type.

A program in PigLatin consists of a sequence of commands which are executed one after another and perform only a single data manipulation. Such a step is fairly high-level, like grouping, joining, or filtering. These operators resemble relational algebra and allow Pig to use traditional database optimizations. First, the program is translated into a logical plan. This plan is optimized and enables Pig to be used with alternative platforms other than Hadoop. In addition, joins are automatically translated into MapReduce plans without the need to manually define and hand-optimize them.

Finally, the logical plan is translated into a MapReduce plan and executed on the underlying platform. With this design, the programmer can use an SQL-like approach to design the application and still easily include and influence custom functions.

Hive. Hive [43,44] organizes data in a structured, relational data model and stores meta information about defined tables. In that aspect Hive closely follows a parallel database approach, still tables and their partitions are stored in a

distributed file system along with the meta information. Along with Hive comes an SQL-like query language called HiveQL. Although HiveQL includes support for equality joins, the language is not able to handle Cartesian products. The programmer can circumvent this problem by extending the language with custom scalar functions, aggregations, and table functions. The user may also include custom mappers or reducers if the possibilities of HiveQL are still insufficient.

Jaql. Another high-level language designed by IBM Research is Jaql [45,46]. Its data model is based on JavaScript Object Notation (JSON). A JSON file is organized as an array of JSON objects. Each object contains name/value pairs where each value can again be a JSON object or an atomic type. Using this data model, Jaql is able to handle (semi-)structured data. Provided schema information is used for type checking and to improve performance. Jaql's main unique selling points are the support for higher-order functions and what they call physical transparency. Physical transparency in this context means that Jaql's execution plan is completely expressed in Jaql. This gives the user a lot of control as they are able to enhance the language without modifying the query language and to influence or even pin the execution plan.

Jaql is used in IBM's InfoSphere BigInsights product [47].

3.3 Dryad

Microsoft Research's Dryad [48], released in 2007, is a parallel execution engine for data flows designed to run on a large cluster of shared-nothing commodity servers. A Dryad query is defined as a directed acyclic graph (DAG) where vertices contain arbitrary, sequential user-code and edges define the data flow through the graph. The programmer can design analysis tasks by wiring up vertices with any number of edges. A nice feature of Dryad DAGs is the support for merging which allows the programmer to create graphs for specific tasks and link or merge them to create more complex tasks.

The Dryad execution engine handles the parallel execution transparently using data parallelism. Dryad uses a job manager process to orchestrate and distribute vertices among the worker machines. The vertices are cloned to multiple machines and each vertex takes a subset of the input. By wiring the vertices in a 1:1, 1:N or M:N fashion using files, TCP pipes, or shared-memory channels any semantics can be expressed. Serialization/deserialization of data for transport is completely left to the programmer. Dryad only supplies her with a small set of library item types such as tuples or newline-terminated strings. This gives the programmer complete control but increases development time and effort. Microsoft Research developed two approaches to ease the integration of Dryad into applications, DryadLINQ and SCOPE.

Recently, Microsoft has decided to focus on Hadoop as opposed to Dryad [49].

DryadLINQ. DryadLINQ [50] is designed to hide the complexity of Dryad from the programmer by integrating it into a programming language. DryadLINQ

bridges .NET's LINQ (Language INtegrated Queries) environment [51] and Dryad. This tightly integrates the queries into one of the managed languages of the .NET environment in contrast to writing MapReduce jobs or defining a query in SQL. These LINQs are translated into Dryad DAGs and executed in parallel.

Apart from the integrated operators, DryadLINQ offers the possibility to add custom operators. The drawback of these operators is that if no information about a possible parallelization is available - they can be supplied by user annotations -, the custom operator will employ only one machine for data processing.

SCOPE. Structured Computations Optimized for Parallel Execution (SCOPE) [52] is a declarative scripting language hiding parallelism from the programmer. SCOPE's design was heavily influenced by SQL and SCOPE adopts operators like (inner / outer) joins, aggregations, selections, and projections. SCOPE can be easily extended by user-defined functions and operators which are categorized into four kinds: extractors to create rows from files, processors for row-wise processing, reducers for processing of groups akin to MapReduce's reduce operation, and combiners to combine two inputs. Contrary to SQL, programmers can not only use a declarative approach to design their programs, but also express them as a series of data transformation steps, i.e., using an imperative approach.

3.4 Hyracks

The computer science group of the University of California, Irvine developed their data parallel platform for data-intensive computations, Hyracks [53]. It features a parallel data flow execution model. Jobs are specified as directed acyclic graphs (DAGs) composed of operators, represented as nodes, and connectors, represented as edges. Operators consist of one or more activities to further distinguish between different operator steps, e.g., a hash join is split into a build and a probe activity connected by a dependency. The probe step can only be executed when the build phase has completed. This further distinction of operators into activities provides Hyracks with more information about the real execution of the overall application. Connectors partition the output for the following operators.

Hyracks abstracts the MapReduce data model and uses tuples with fields of arbitrary types. The design of Hyracks emphasizes extensibility. The library of Hyracks provides a set of reusable operators and connectors but the programmer can extend the operators, connectors, and types by providing implementations.

The execution engine of Hyracks is based on a master-worker scheme, where each node controller registers at the cluster controller. The cluster controller accepts jobs and then schedules the execution.

For the execution, the operators are decomposed into their activities. The activities are split into stages. Each stage comprises the activities that can be pipelined. The activities are multiplied between nodes and then executed stage by stage using a lazy execution model.

Hyracks features the Asterix Data Model (ADM) which supports nested data types similar to JSON.

On top of Hyracks is the Asterix Query language (AQL) [54]. AQL is inspired by XQuery [55] but eliminated the XML-specific and document-specific features.

3.5 Stratosphere

A research team comprised of TU Berlin, HU Berlin, and HPI Potsdam has since 2008 been building the Stratosphere [56] system, consisting of the PACT programming model and the massively parallel execution engine Nephele.

The PACT programming model is a generalization of the MapReduce programming model and operates on a key/value data model. A Parallelization Contract (PACT) consists of one input contract, optionally one output contract, and a user-defined first-order function. An input contract is a second-order function defining how the data is passed to the user-defined function, e.g., a Map input contract would simply pass each key/value pair to a different call of the user-defined function while a Reduce input contract would pass all key/value pairs sharing the same key to a call of the user-defined function. In contrast to the MapReduce programming model, Stratosphere's PACTs support multiple inputs to facilitate operations such as joins.

Output contracts are annotations to a PACT data analysis program to describe properties of the user-defined function, e.g., the key of the input key/value pairs is not modified. This provides the cost-based optimizer with additional information, e.g., the key/value pairs would need no repartitioning after the key, if they were partitioned based on the requirements of a preceding PACT which has an output contract that preserves the keys.

A program is created by chaining different PACTs together. Such a program is then passed to the optimizer and translated into a Nephele DAG. These DAGs are similar to Dryad DAGs, vertices contain user-code and the edges represent channels for data transport. By multiplying vertices, scheduling them among multiple nodes, and wiring them with channels, according to the required input contract, the execution is parallelized.

In contrast to MapReduce, Nephele allows for communication via network and memory channels in addition to disk channels, thus enabling better support for low latency (streaming) queries and multi-core environments.

4 Vertical Scaling

Vertical scaling includes assembling machines with more memory and higher performing CPUs as well as the application of specialized hardware like GPUs as co-processors. The idea of incorporating co-processors to accelerate database systems is not new, already in 1978 D.J. DeWitt proposed using multiprocessors to accelerate query execution [57]. At that time, CPU development was still incredibly fast so that using co-processors was quickly overtaken by newer CPU generations. Today, with the ILP wall, the memory wall, and the power wall, using co-processors looks more promising.

In this section, we will introduce modern hardware architectures and discuss their application for Big Data Analytics.

4.1 FPGA

Before the development of programmable logic, hardware designers were forced to design logic circuits at the board level using standard components or application specific integrated circuits (ASIC). In 1956, Wen Tsing Chow invented programmable read-only memory (PROM), memory that could be programmed after manufacturing by burning some of the fuses and effectively setting these bits to zero [58]. The development of PROMs was the foundation of the first PLAs (Programmable Logic Arrays) in 1978. These chips contain programmable AND-planes wired to a programmable OR-plane which could implement functions in the Sum of Products form. Later, PALs (Programmable Array Logics) were developed which only had a programmable AND-plane followed by a hard-wired OR-plane. The advantages of PALs were the cheaper production and the faster logic due to the hard-wired OR-plane. PALs and PLAs are grouped together as Simple Programmable Logic Devices (SPLDs). To answer the technological demand, Complex Programmable Logic Devices (CPLDs) were developed. CPLDs consist of SPLDs combined into a single chip with programmable interconnects. To further increase capacity, Mask-programmable Gate Arrays (MPGAs) combine arrays of transistors that can be freely wired in the factory. This introduced high costs and a long development phase for hardware manufacturers but they motivated the design of Field-Programmable Gate Arrays (FPGAs). The first FPGA was patented in 1984 and one year later created by Ross Freeman and Bernard Vonderschmitt. The development of FPGAs changed the way of designing digital circuits. The ability to program the chips in the field as often as needed allows hardware developers to do fast prototyping which also results in shorter time to market. In addition, error correction no longer needs a redesign of the board as it can be simply specified in the configuration [59,60,61].

One of the drawbacks of FPGAs is the lower frequency at which they operate and the low-level design of applications. Today, FPGAs are programmed using circuit schematics or by using a hardware description language like VHDL or Verilog [62] but the task of programming is still fairly low-level.

Architecture. FPGAs are arrays of configurable logic blocks (CLBs) that are connected through routing channels and surrounded by programmable Input/Output Blocks, i.e., interfaces between the FPGA and outside resources. Multiple logic cells and a switch box form a CLB. Switch boxes connect the logic cells to the interconnect fabric. A logic cell contains logic gates, carry logic and a storage element. A logic gate is implemented as a Lookup Table (LUT), an n-LUT encodes an n-input boolean function as truth table. As storage elements, D flip flops or latches are being used. Nowadays, FPGAs may also contain hard intellectual property (IP) cores - frequently used functionality as discrete chips - such as block RAM (BRAM), multiplier units, or PowerPC cores [59,60,61,63,62].

Applications. J. Teubner et al. showed that FPGAs can be used for data processing with the use case of a median operator based on sorting networks [63]. When processing multiple streams in parallel, FPGAs grant a performance benefit over CPUs because of cache misses. In addition, FPGAs have a lower power consumption than CPUs [63,62], which becomes increasingly important today.

Another application is shown by Mitra et al. [64]. They use FPGAs to filter XPath queries in a publisher-subscriber system with an order of magnitude performance gain.

Microsoft Research tried to hide the FPGA programming from the user with their system Kiwi [65]. It uses C# parallel programming language constructs and translates them into Verilog, thus making FPGAs more accessible to programmers of higher-level languages.

Apart from research, FPGAs are employed in several data analytics companies to accelerate query execution. IBM's Netezza [66] uses FPGAs for data filtering, Kickfire [67] accelerates SQL operations in MySQL, and XtremeData with their database analysis appliance (dbX) [68] uses FPGAs to accelerate SQL operators, for data movement capabilities, and statistics gathering.

4.2 GPU

GPUs started out as graphics accelerators. In 2000, GPUs became more programmable and researchers started to use graphics adapters for non-graphics applications. Shortly thereafter, the GPGPU⁷ (General-Purpose computation on Graphics Processing Units) movement started [70]. At this time, the computational power of GPUs was only available to those familiar with graphics programming languages, e.g., OpenGL or Direct3D. Programming for the GPU was complicated and required in-depth knowledge of the architecture. In 2004, Ian Buck et al. from Stanford University published Brook, "a system for general-purpose computation on graphics hardware" [71]. Brook was designed as a C extension and made GPU processing available for a wider audience. Ian Buck joined NVIDIA and in 2006, they released the first version of CUDA (Compute Unified Device Architecture), a parallel computing platform and programming model for general-purpose algorithms [70].

Late 2008, the Khronos Group released OpenCL 1.0 (Open Computing Language), an open, cross-platform parallel programming model [72] and therefore a competitor to CUDA. In the same year, Microsoft introduced DirectCompute at Gamefest 2008, an API for GPGPU on Microsoft Windows Vista and Windows 7 as part of DirectX 11 [73].

Today, clusters with GPUs are employed throughout the world and three out of the top five supercomputers of the world integrate GPUs for increased processing power [74].

⁷ In 2002, Mark Harris coined the term GPGPU [69].

Architecture. Nowadays, graphics adapters are connected to RAM via PCIe v2.x x16 bus featuring a theoretical bandwidth of 8 GB/s in each direction [75]. PCI-SIG released the PCIe 3.0 standard late 2010 [75], which will double the available bandwidth. Currently (early 2012), only the newest graphics adapters support PCIe 3.0. The architecture of devices differs between vendors but the general abstract view can be easily transferred. For brevity, we will refer only to NVIDIA's Fermi architecture [76] and server-grade graphics adapters like NVIDIA's Tesla series [77].

A graphics adapter contains multiple streaming multiprocessors (SM). Each SM consists of several⁸ scalar processors, each featuring a fully pipelined integer arithmetic logic unit (ALU) and floating point unit (FPU) to serve both integer and floating point performance.

On the device, NVIDIA employs a hierarchical memory model. The biggest memory is the global device memory (up to 6 GB) which all SMs can access. Global memory is also used to exchange data between RAM and device via the PCIe bus. Between global memory and the processors is a L2 cache (768 kB). Next are 64 kB of configurable memory which can be split between shared memory and L1 cache⁹. This memory is bound to a SM and accessible only by the scalar processors within an SM. Additionally, each SM is equipped with a set of registers (32,768 x 32-bit) which are distributed among the scalar processors by need. Like in the CPU cache hierarchy, the closer the memory is located to the processors, the smaller and faster they are.

The architectural problems of GPUs are the relatively small device memory of up to 6 GBs in comparison to main memory systems with up to 2 TB of RAM, and that graphics adapters can process only locally available data, i.e., all data needs to be copied to the device over the comparably slow PCIe bus before processing. Once the data is on the device, the device memory excels with a bandwidth of 144 GB/s [77]. When designing a GPGPU application, these bottlenecks have to be taken into account as they may incur costs that make the implementation on a GPU non-competitive in comparison to a CPU implementation.

Programming Model. A program for GPUs is divided into two parts: host code and kernels. The host code allocates resources, sets parameters, and launches kernels. A kernel is a program that is compiled and afterwards executed on the GPU, and specifies the commands for a thread, i.e., all threads in a kernel perform the same task and only differentiate in which parts of data are processed. These threads are grouped into blocks. Therefore, a kernel is a grid of blocks, where each block consists of up to 1024 threads. A block is assigned to a SM and each thread of a block is evaluated by a scalar processor in the SM. The execution is split into warps, groups of 32 threads, that run at the same time. This hierarchy is important, as it reflects the different synchronization possibilities: Threads of a block may synchronize using a barrier and can share data stored in shared memory. Threads of different blocks are completely isolated from each

⁸ 32 to 48 processors in modern graphics adapters.

⁹ 16 kB shared memory / L1 cache is the minimum assigned to each.

other apart from access to global memory. Therefore, the programmer cannot assume any order among the threads.

To write a good performing kernel, the programmer has to meet a few challenges:

- Graphics devices only offer limited synchronization possibilities.
- Dynamic memory allocation inside a kernel is not possible.
- Memory accesses should be coalesced¹⁰ to fully utilize the high bandwidth of device memory.
- Divergence at warp level should be avoided.

Divergence in this context means that two threads take different paths when evaluating a condition. The SM has only a single program counter, therefore all threads on a SM must execute the same instruction per cycle. If threads take different code paths, the execution is serialized and part of the processors idle. As only one warp at a time is executed, inter-warp divergence poses no problem. This execution is also called SIMT (Single Instruction, Multiple Threads) which in contrast to SIMD (Single Instruction, Multiple Data) allows each thread to branch or have it's own register [78]. This is different from SIMD instructions like on the CPU, e.g., on the CPU a SIMD instruction may take a 128 bit input and process it using the same function as four 32 bit words, a single instruction for multiple data.

Applications. GPUs yield a lot of computing power, however a lot of problems in database systems and Big Data Analytics analyze massive amounts of data. Therefore, the most required resources are available memory and bandwidth. In that aspect, GPUs seem to have no place in this scenario, but there are advantages to the usage of GPUs if we focus on smaller tasks. For the GPU to provide any benefits, the data shipped to the device needs to be small, the performed task must be computation intensive, or the shipped data has to be reused multiple times to amortize the slow transfer over the PCIe bus.

Numerous algorithms for sorting have been published using different methods like sorting networks [79], sample sort [80], merge sort [81], and radix sort [82][83]. The number of sorted keys per second on GPUs is generally higher than on CPUs, but only if the data transfer over PCIe is excluded [84].

B. He et al. investigated relational database operators on GPUs in "GDB", a hybrid CPU & GPU query processor embedded into their own small database system [85][86]. They designed a co-processing scheme, in which the work is distributed among CPU, GPU, or both, using cost models on a per operator basis. They noted a 2x-7x speedup for complex queries like joins and are slower by a factor of 2x-4x for simple operators like selections when using only the GPU operations [85]. In addition, the operators are limited to device memory. If the

¹⁰ Coalesced memory accesses describe a pattern, where each thread reads data neighboring to the data of its predecessor, i.e., the threads access a consecutive block of memory without any gaps.

input exceeds device memory and partitioning of the input becomes necessary, the GPU would lose performance dramatically.

A different kind of application suitable for GPUs are index lookups. Creating an index and storing it on the GPU to answer requests avoids passing big amounts of data continuously over the PCIe bus. The result of an index lookup is usually highly selective and returns only a small number of result tuples. There have been different suggestions for index structures like Cache-sensitive search (CSS) trees [87,86], indexes applying hierarchical blocking calibrated to the SIMD width of GPUs [88], or speculative evaluated prefix trees [89]. They can leverage the power of GPUs by issuing multiple queries at once [88] or by speculatively evaluating all parts of an index [89]. Which approach to choose depends on the problem to be solved.

In effect, research puts a lot of effort into exploiting the computational power of GPUs for database systems but the PCIe bottleneck and the small device memory remain hard problems for data-intensive applications. Further generations of GPUs will probably increase the available device memory and with the introduction of PCIe 3.0 the data transfer bottleneck will be revised. This may lead to a re-evaluation of GPUs and might make them more attractive to commercial systems. Up to now, only a single commercial database system, ParStream [38], exploits GPUs for query processing.

4.3 APU

AMD released the first Accelerated Processing Units (APUs) named Fusion at the Computex conference in 2010 [90]. One objective was to raise competition for Intel, which dominated the netbook market with their low-power and at the same time good performing Atom CPU. This was due to Intel's architects integrating controllers and graphics processors onto the same die [91]. A different objective was the integration of GPU cores onto a CPU die [92].

The integration of GPU cores eliminates the two main issues of graphics adapters - PCIe transfer and small memory size - as the GPUs or vector units directly access the main memory, albeit the memory is currently split into regions for each core type. With such an architecture, applications can extract the computing power of the GPU cores using DirectCompute or OpenCL. AMD regards this development to merge heterogeneous processing cores onto the same die as the future of hardware development where each task is handled by the hardware best fitting the problem. Apparently, Sandia National Labs share this opinion as they installed the first HPC Cluster with AMD Fusion chips to evaluate this heterogeneous processor model [93]. At this point in time (2012), only notebook or desktop APUs are available so that little work has been conducted on APUs in the context of Big Data Analytics. AMD proclaims that server-grade APUs will be delivered "in the years ahead and when the time is right" [94]. It will be interesting to see how AMD migrates the processing power from GPUs for server-grade APUs as some parameters change: Main memory is bigger than device memory but at the same time, RAM is slower by a factor of about five compared to the currently used device memory. Will this starve the GPU cores or

introduce memory latencies that cannot be hidden by the threading model? Until AMD publishes information about a server-grade die, the evaluation of APUs in the context of DBMSs or Big Data Analytics will remain a lab experiment to analyze heterogeneous processing.

4.4 MIC

Intel's Many Integrated Core (MIC¹¹) architecture [95] builds upon the results of three research projects: the many-core visual computing project Larrabee, the 80-core Tera-scale research chip program, and the Single-chip Cloud Computer (SCC) initiative.

Larrabee is meant for visual computing as competitor to NVIDIA's and AMD's GPUs. It consists of multiple in-order x86 CPUs augmented with 16-wide vector processing units (VPU). Each core has a L1 cache and a subset of a local coherent 2nd level cache. A bi-directional ring network connects the CPU cores, memory & I/O controllers, and fixed function logic blocks with a 512-bits per direction bus [96].

In the Tera-scale research program, Intel researches energy efficient designs for multi-core chips. A research prototype chip comprised of 80 simple cores, each containing two floating point engines, features one teraflop of computing power [97].

Intel Labs Single-chip Cloud Computer consists of 24 tiles with two cores each. Each tile is equipped with a router connecting the tile with other cores and four DDR3 memory controllers. The tiles are organized into four regions. One region of tiles is mapped to a specific memory controller. Each core has an on-chip SRAM for message-passing and a private portion of the external DRAM. By default, most of the DRAM is used for private memory and only a small fraction is used as shared memory for all cores. The SCC has no hardware cache coherence support among cores although each core has two levels of cache. The intention is to encourage the use of distributed memory software models using the hardware message-passing support [98,99].

One result of these three research programs is the MIC architecture called Knights Corner which is aimed at High Performance Computing (HPC). Knights Corner will integrate more than 50 cores onto a single chip [95]. The first presentation of Intels accelerator, the first silicon of the "Knights Corner", was at the International Supercomputing Conference 2011 [100]. The accelerator delivers more than 1 TFLOPS of double precision floating point performance and will be available for commercial use. Its predecessor, "Knights Ferry", was only available as software development platform for co-processors to selected development partners.

An important benefit of Knights Corner is that it uses the same programming environment as Intel Xeon CPUs. Therefore, applications can be easily ported [95].

¹¹ pronounced "Mike".

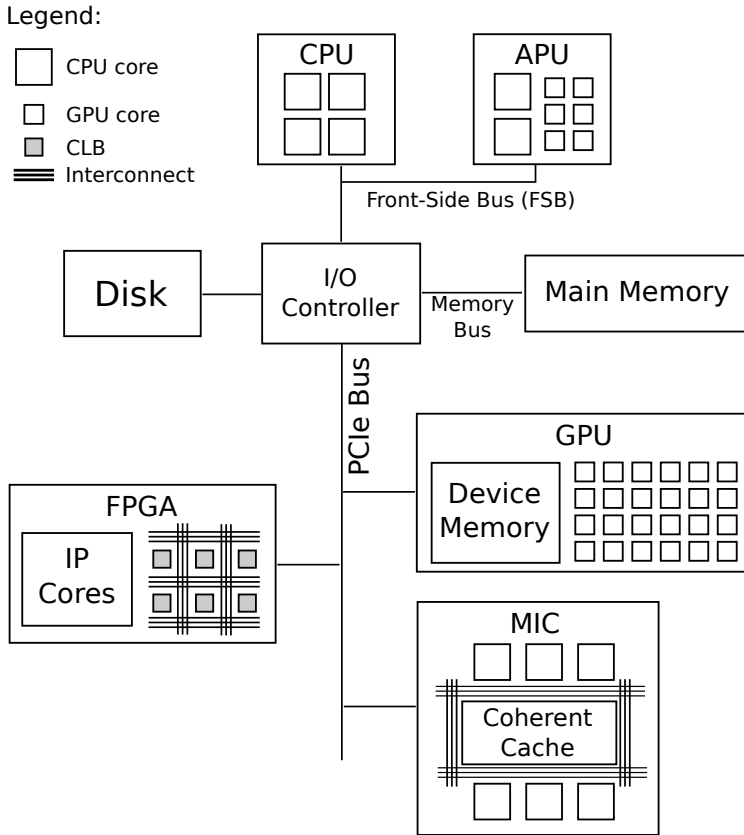


Fig. 4. Overview of a highly heterogeneous system incorporating the presented architectures

Whether Knights Corner is suitable for database operations and how the exact architecture specification is, cannot be said at this point as the Knights Corner has not been released yet.

4.5 Summary

Figure 4 shows a heterogeneous system consisting of the discussed architectures. On the top, the CPU features multiple cores designed to execute arbitrary sequential programs with limited parallelism. The APU to the right, consists of CPU cores, too, but in addition, several GPU cores are added to increase the computational power for parallel problems. To address the additional computing resources, the programmer must fit the program design to a parallel programming framework like OpenCL. Like CPUs, the APUs can directly access the main memory without passing through a low-bandwidth bus like PCIe.

The other devices need to ship the data over the PCIe bus to start processing and ship the results back to make them available to other components. On the

left, the FPGA offers multiple CLBs and possibly hard IP cores to tailor the execution to the problem at hand. Even with a lower frequency than CPUs or GPUs, the fine-grained configuration allows FPGAs to process data efficiently. The drawback is the comparatively low-level programming of FPGAs in circuit designs or languages like Verilog or VHDL.

On the right, the GPU offers fast device memory once the data is on the device. Along with the high amount of processing units, the GPUs can process parallel problems with high computational and bandwidth requirements. The limitations of GPUs are the reduced support for branching and the small amount of device memory.

The MIC architecture offers a similar programming framework to CPUs and, therefore, provides good portability of existing programs. The use of CPU cores allows for more complex branching and control structures than in GPUs.

5 Conclusion

Today, a developer has a lot of systems available for Big Data Analytics and a broad range of co-processors. The choice of system depends on the specific scenario at hand. If the data set is extremely big, single machines may not be able to handle the data load. Therefore, using one of the massive parallel analysis systems may be mandatory. In the case of smaller data sets, the benefits of such a system may not compensate for the overhead of managing a cluster.

Another important factor is the intended workload. While distributed systems are good to analyze huge amounts of data with aggregations and big results, they suffer in comparison to specialized systems in terms of efficiency. Lastly, it might also depend on the existing knowledge of programming models and languages of the developer if productivity is concerned.

The interesting point is how these systems will develop in the future apart from gaining features or changing algorithms. If we take a look at the programming models of distributed systems and modern hardware architectures, we see that they both employ a parallel model. The combination of distributed systems with co-processors on specific nodes may bring additional performance, but requires the systems to be hardware-aware. The integration of accelerators benefits from the application of parallel programming models to distribute tasks as they can be easily translated to the parallel models of the hardware. An example for this is Mars [101], a MapReduce framework, which integrates GPU processing into Hadoop. Mars eases the GPU programming by focusing on the well-known MapReduce programming model. Still, the map and reduce implementations for the CPU and the GPU have to be provided. This highlights, that a system addressing a heterogeneous cluster - combining horizontal and vertical scaling - proves to be challenging. On the one hand, the system needs to be general enough to allow for the handling of different architectures. On the other hand, the burden on the user should be as low as possible. Ideally, the user should maintain only a single operator implementation that can be ported to different architectures. One possibility to achieve this would be to use a framework like OpenCL and integrate it into a distributed system.

In the future, CPUs will continue to gain more cores. CPUs will change into more parallel systems in either of two ways: CPUs will integrate heterogeneous cores - like in AMD's APUs - and present one interface for sequential and control structure heavy code parts and one for data-parallel operators, or they will follow Intel's SCC idea, integrating a lot of cores onto a single die. Whatever the future holds, parallel programming - and especially parallel programming models - will become more important because there is a huge difference between running a sequential program in parallel as opposed to writing a parallel program.

References

1. CERN: Worldwide LHC Computing Grid (December 2011), <http://public.web.cern.ch/public/en/LHC/Computing-en.html>
2. Manyika, J., Chui, M., Brown, B., Bughin, J., Dobbs, R., Roxburgh, C., Hung Byers, A.: Big Data: The Next Frontier for Innovation, Competition, and Productivity (June 2011), http://www.mckinsey.com/Insights/MGI/Research/Technology_and_Innovation/Big_data.The_next_frontier_for_innovation
3. Liu, X., Thomsen, C., Bach Pedersen, T.: The ETLMR MapReduce-Based ETL Framework. In: Bayard Cushing, J., French, J., Bowers, S. (eds.) SSDBM 2011. LNCS, vol. 6809, pp. 586–588. Springer, Heidelberg (2011)
4. Alexandrov, A., Ewen, S., Heimes, M., Hueske, F., Kao, O., Markl, V., Nijkamp, E., Warneke, D.: MapReduce and PACT - Comparing Data Parallel Programming Models. In: Proceedings of the 14th Conference on Database Systems for Business, Technology, and Web, BTW 2011, pp. 25–44. GI, Bonn (2011)
5. Gillick, D., Faria, A., Denero, J.: MapReduce: Distributed Computing for Machine Learning (2006)
6. Ghoting, A., Krishnamurthy, R., Pednault, E., Reinwald, B., Sindhwani, V., Tatikonda, S., Tian, Y., Vaithyanathan, S.: SystemML: Declarative Machine Learning on MapReduce. In: Proceedings of the 2011 IEEE 27th International Conference on Data Engineering, ICDE 2011, pp. 231–242. IEEE Computer Society, Washington, DC (2011)
7. Kang, U., Tsourakakis, C.E., Faloutsos, C.: PEGASUS: Mining Peta-scale Graphs. *Knowl. Inf. Syst.* 27(2), 303–325 (2011)
8. Cohen, J.: Graph Twiddling in a MapReduce World. *Computing in Science Engineering* 11(4), 29–41 (2009)
9. Zhao, W., Ma, H., He, Q.: Parallel *K*-Means Clustering Based on MapReduce. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 674–679. Springer, Heidelberg (2009)
10. The Apache Software Foundation: Applications powered by Hadoop (December 2011), <http://wiki.apache.org/hadoop/PoweredBy>
11. Facebook: Hadoop (December 2011), http://www.facebook.com/note.php?note_id=16121578919
12. Office of Electricity Delivery & Energy Reliability, U.S. Department of Energy: Smart Grid (December 2011), <http://energy.gov/oe/technology-development/smart-grid>
13. Henschen, D.: 12 Top Big Data Analytics Players (December 2011), <http://www.informationweek.com/news/galleries/software/bi/231900870>

14. Amdahl, G.M.: Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In: Proceedings of the April 18-20, 1967, Spring Joint Computer Conference, AFIPS 1967 (Spring), pp. 483–485. ACM, New York (1967)
15. Stoica, I., Morris, R., Liben-Nowell, D., Karger, D., Kaashoek, M.F., Dabek, F., Balakrishnan, H.: Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. *IEEE Transactions on Networking* 11 (February 2003)
16. Computer History Museum: 1965 - “Moore’s Law” Predicts the Future of Integrated Circuits (December 2011), <http://www.computerhistory.org/semiconductor/timeline/1965-Moore.html>
17. Intel Corporation: White Paper: Intel Next Generation Intel Microarchitecture (Nehalem) (2008), http://www.intel.com/pressroom/archive/reference/whitepaper_nehalem.pdf
18. Intel Corporation: Intel Xeon Processor 7500 Series: Product Brief (December 2011), <http://www.intel.com/content/www/de/de/mission-critical/mission-critical-computing-xeon-7500-brief.html>
19. Intel Corporation: Intel Xeon Processor E7-8870 Specification (December 2011), <http://ark.intel.com/products/53580/Intel-Xeon-Processor-E7-8870-%2830M-Cache-2.40-GHz-6.40-GTs-Intel-QPI%29>
20. Advanced Micro Devices, Inc.: AMD Opteron 6282 SE Specification (December 2011), <http://products.amd.com/en-us/OpteronCPUdetail.aspx?id=756>
21. Codd, E.F.: A Relational Model of Data for Large Shared Data Banks. *Commun. ACM* 13, 377–387 (1970)
22. Astrahan, M.M., Blasgen, M.W., Chamberlin, D.D., Eswaran, K.P., Gray, J.N., Griffiths, P.P., King, W.F., Lorie, R.A., McJones, P.R., Mehl, J.W., Putzolu, G.R., Traiger, I.L., Wade, B.W., Watson, V.: System R: Relational Approach to Database Management. *ACM Trans. Database Syst.* 1, 97–137 (1976)
23. Held, G.D., Stonebraker, M.R., Wong, E.: INGRES: A Relational Data Base System. In: Proceedings of the May 19-22, 1975, National Computer Conference and Exposition, AFIPS 1975, pp. 409–416. ACM, New York (1975)
24. Copeland, G.P., Khoshafian, S.N.: A Decomposition Storage Model. In: Proceedings of the 1985 ACM SIGMOD International Conference on Management of Data, SIGMOD 1985, pp. 268–279. ACM, New York (1985)
25. Boncz, P.A., Kersten, M.L., Manegold, S.: Breaking the Memory Wall in MonetDB. *Communications of the ACM* 51(12), 77–85 (2008)
26. Boncz, P.A., Manegold, S., Kersten, M.L.: Database Architecture Optimized for the New Bottleneck: Memory Access. In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 54–65. Morgan Kaufmann Publishers Inc., San Francisco (1999)
27. Ailamaki, A., DeWitt, D.J., Hill, M.D., Wood, D.A.: DBMSs on a Modern Processor: Where Does Time Go? In: Proceedings of the 25th International Conference on Very Large Data Bases, VLDB 1999, pp. 266–277. Morgan Kaufmann Publishers Inc., San Francisco (1999)
28. Plattner, H., Zeier, A.: In-Memory Data Management: An Inflection Point for Enterprise Applications. Springer (2011)
29. Stonebraker, M., Abadi, D.J., Batkin, A., Chen, X., Cherniack, M., Ferreira, M., Lau, E., Lin, A., Madden, S., O’Neil, E.J., O’Neil, P.E., Rasin, A., Tran, N., Zdonik, S.B.: C-Store: A Column-oriented DBMS. In: Böhm, K., Jensen, C.S., Haas, L.M., Kersten, M.L., Larson, P.K., Ooi, B.C. (eds.) VLDB, pp. 553–564. ACM (2005)

30. Vertica: Vertica (December 2011), <http://www.vertica.com/>
31. Actian Corporation: Vectorwise (December 2011), <http://www.actian.com/products/vectorwise>
32. Fushimi, S., Kitsuregawa, M., Tanaka, H.: An Overview of the System Software of a Parallel Relational Database Machine GRACE. In: Proceedings of the 12th International Conference on Very Large Data Bases, VLDB 1986, pp. 209–219. Morgan Kaufmann Publishers Inc., San Francisco (1986)
33. DeWitt, D.J., Gerber, R.H., Graefe, G., Heytens, M.L., Kumar, K.B., Muralikrishna, M.: GAMMA - A High Performance Dataflow Database Machine. In: Proceedings of the 12th International Conference on Very Large Data Bases, VLDB 1986, pp. 228–237. Morgan Kaufmann Publishers Inc., San Francisco (1986)
34. Teradata Corporation: Teradata (December 2011), <http://www.teradata.com/>
35. EMC Corporation: Greenplum (December 2011), <http://www.greenplum.com>
36. Teradata Corporation: Aster Data (December 2011), <http://www.asterdata.com/>
37. Friedman, E., Pawlowski, P., Cieslewicz, J.: SQL/MapReduce: A Practical Approach to Self-describing, Polymorphic, and Parallelizable User-defined Functions. Proc. VLDB Endow. 2, 1402–1413 (2009)
38. empulse GmbH: ParStream (December 2011), <http://www.parstream.com>
39. Ghemawat, S., Gobioff, H., Leung, S.T.: The Google File System. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles, SOSP 2003, pp. 29–43. ACM, New York (2003)
40. The Apache Software Foundation: Welcome to Hadoop! (December 2011), <http://hadoop.apache.org>
41. Olston, C., Reed, B., Srivastava, U., Kumar, R., Tomkins, A.: Pig Latin: A Not-So-Foreign Language for Data Processing. In: SIGMOD 2008: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, pp. 1099–1110. ACM, New York (2008)
42. The Apache Software Foundation: Welcome to Apache Pig! (December 2011), <http://pig.apache.org>
43. Thusoo, A., Sarma, J.S., Jain, N., Shao, Z., Chakka, P., Anthony, S., Liu, H., Wyckoff, P., Murthy, R.: Hive: A Warehousing Solution Over a Map-Reduce Framework. Proc. VLDB Endow. 2, 1626–1629 (2009)
44. The Apache Software Foundation: Welcome to Hive! (December 2011), <http://hive.apache.org>
45. Jaql - Query Language for JavaScript Object Notation (JSON) (December 2011), <http://code.google.com/p/jaql/>
46. Beyer, K.S., Ercegovac, V., Gemulla, R., Balmin, A., Eltabakh, M., Kanne, C.C., Ozcan, F., Shekita, E.J.: Jaql: A Scripting Language for Large Scale Semistructured Data Analysis. In: PVLDB 2011, pp. 1272–1283 (2011)
47. IBM: InfoSphere BigInsights (December 2011), <http://www-01.ibm.com/software/data/infosphere/biginsights/features.html>
48. Isard, M., Budiu, M., Yu, Y., Birrell, A., Fetterly, D.: Dryad: Distributed Data-Parallel Programs from Sequential Building Blocks. In: EuroSys 2007: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007, pp. 59–72. ACM, New York (2007)
49. Microsoft Corporation: The Windows HPC Team Blog (November 2011), <http://blogs.technet.com/b/windowshpc/archive/2011/11/11/hpc-pack-2008-r2-sp3-and-windows-azure-hpc-scheduler-released.aspx>

50. Yu, Y., Isard, M., Fetterly, D., Budiuh, M., Erlingsson, U., Gunda, P.K., Currey, J.: DryadLINQ: A System for General-Purpose Distributed Data-Parallel Computing Using a High-Level Language. In: Draves, R., van Renesse, R. (eds.) OSDI, pp. 1–14. USENIX Association (2008)
51. Microsoft Research: The LINQ project (December 2011), <http://msdn.microsoft.com/en-us/library/bb397926.aspx>
52. Chaiken, R., Jenkins, B., Larson, P.A., Ramsey, B., Shakib, D., Weaver, S., Zhou, J.: SCOPE: Easy and Efficient Parallel Processing of Massive Data Sets. Proc. VLDB Endow. 1, 1265–1276 (2008)
53. Borkar, V.R., Carey, M.J., Grover, R., Onose, N., Vernica, R.: Hyracks: A Flexible and Extensible Foundation for Data-intensive Computing. In: ICDE, pp. 1151–1162 (2011)
54. Behm, A., Borkar, V.R., Carey, M.J., Grover, R., Li, C., Onose, N., Vernica, R., Deutsch, A., Papakonstantinou, Y., Tsotras, V.J.: ASTERIX: Towards a Scalable, Semistructured Data Platform for Evolving-world Models. Distrib. Parallel Databases 29, 185–216 (2011)
55. XQuery 1.0: An XML Query Language (December 2011), <http://www.w3.org/TR/xquery/>
56. Battré, D., Ewen, S., Hueske, F., Kao, O., Markl, V., Warneke, D.: Nephele/PACTs: A Programming Model and Execution Framework for Web-Scale Analytical Processing. In: Proceedings of the 1st ACM Symposium on Cloud Computing, SoCC 2010, pp. 119–130. ACM, New York (2010)
57. DeWitt, D.J.: DIRECT - A Multiprocessor Organization for Supporting Relational Data Base Management Systems. In: Proceedings of the 5th Annual Symposium on Computer Architecture, ISCA 1978, pp. 182–189. ACM, New York (1978)
58. Downes-Powell, G.: What is a PROM Chip? (December 2011), http://www.ehow.com/info_10005464_prom-chip.html
59. FPGA Central: History of the Programmable Logic (December 2011), <http://www.fpgacentral.com/docs/fpga-tutorial/history-programmable-logic>
60. Brown, S., Rose, J.: Architecture of FPGAs and CPLDs: A Tutorial. IEEE Design and Test of Computers 13, 42–57 (1996)
61. EngineersGarage: Field Programmable Gate Array (FPGA) (December 2011), <http://www.engineersgarage.com/articles/fpga-tutorial-basics>
62. Mueller, R., Teubner, J., Alonso, G.: Data Processing on FPGAs. Proc. VLDB Endow. 2, 910–921 (2009)
63. Mueller, R., Teubner, J., Alonso, G.: Sorting Networks on FPGAs. The VLDB Journal, 1–23, doi:10.1007/s00778-011-0232-z
64. Mitra, A., Vieira, M.R., Bakalov, P., Tsotras, V.J., Najjar, W.A.: Boosting XML Filtering Through a Scalable FPGA-based Architecture. In: CIDR (2009)
65. Greaves, D., Singh, S.: Kiwi: Synthesis of FPGA Circuits from Parallel Programs. In: 16th International Symposium on Field-Programmable Custom Computing Machines, FCCM 2008, pp. 3–12 (April 2008)
66. Netezza (December 2011), <http://www.netezza.com/data-warehouse-appliance-products/index.aspx>
67. Kickfire (December 2011), <http://www.kickfire.com/>
68. Scofield, T., Delmerico, J., Chaudhary, V., Valente, G.: XtremeData dbX: An FPGA-Based Data Warehouse Appliance. Computing in Science Engineering 12(4), 66–73 (2010)

69. GPGPU.org: About GPGPU (December 2011), <http://gpgpu.org/about>
70. NVIDIA: CUDA: Parallel Programming Made Easy (December 2011), http://www.nvidia.com/object/cuda_home_new.html
71. Buck, I., Foley, T., Horn, D., Sugerman, J., Fatahalian, K., Houston, M., Hanrahan, P.: Brook for GPUs: Stream Computing on Graphics Hardware. *ACM Transactions on Graphics* 23, 777–786 (2004)
72. Khronos Group: The Khronos Group Releases OpenCL 1.0 Specification (December 2011), http://www.khronos.org/news/press/the_khronos_group_releases_openc1.1.0_specification
73. Microsoft: DirectX 11 DirectCompute: A Teraflop for Everyone (December 2011), <http://www.microsoft.com/download/en/details.aspx?displaylang=en&id=16995>
74. Top500.org: Top 500 Supercomputers (November 2011), <http://www.top500.org/list/2011/11/100>
75. PCI-SIG: PCIe 3.0 FAQ (December 2011), http://www.pcisig.com/news_room/faqs/pcie3.0_faq/#EQ2
76. NVIDIA: NVIDIAs Next Generation CUDA Compute Architecture: Fermi (December 2011), http://www.nvidia.com/content/PDF/fermi_white_papers/NVIDIA_Fermi_Compute_Architecture_Whitepaper.pdf
77. NVIDIA: NVIDIA Tesla C2075 (December 2011), <http://www.nvidia.com/docs/I0/43395/NV-DS-Tesla-C2075.pdf>
78. NVIDIA: NVIDIA CUDA C Programming Guide (2011)
79. Govindaraju, N., Gray, J., Kumar, R., Manocha, D.: GPUteraSort: High Performance Graphics Co-processor Sorting for Large Database Management. In: *Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, SIGMOD 2006*, pp. 325–336. ACM, New York (2006)
80. Leischner, N., Osipov, V., Sanders, P.: GPU Sample Sort. In: *24th IEEE International Symposium on Parallel and Distributed Processing (IPDPS)*, pp. 1–10 (2010)
81. Satish, N., Kim, C., Chhugani, J., Nguyen, A.D., Lee, V.W., Kim, D., Dubey, P.: Fast Sort on CPUs and GPUs: A Case for Bandwidth Oblivious SIMD Sort. In: *Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010*, pp. 351–362. ACM, New York (2010)
82. Merrill, D.G., Grimshaw, A.S.: Revisiting Sorting for GPGPU Stream Architectures. Technical Report CS2010-03, University of Virginia, Department of Computer Science, Charlottesville, VA (2010)
83. Satish, N., Harris, M., Garland, M.: Designing Efficient Sorting Algorithms for Manycore GPUs. In: *Proceedings of the 2009 IEEE International Symposium on Parallel & Distributed Processing, IPDPS 2009*, pp. 1–10. IEEE Computer Society, Washington, DC (2009)
84. Wassenberg, J., Sanders, P.: Faster Radix Sort via Virtual Memory and Write-Combining. *CoRR* abs/1008.2849 (2010)
85. He, B., Lu, M., Yang, K., Fang, R., Govindaraju, N., Luo, Q., Sander, P.: Relational Query Coprocessing on Graphics Processors. *ACM Transactions on Database Systems (TODS)* 34(4), 21 (2009)
86. He, B., Yang, K., Fang, R., Lu, M., Govindaraju, N., Luo, Q., Sander, P.: Relational Joins on Graphics Processors. In: *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, pp. 511–524. ACM (2008)
87. Rao, J., Ross, K.A.: Cache Conscious Indexing for Decision-Support in Main Memory. In: *Proceedings of the 25th International Conference on Very Large Data*

- Bases, VLDB 1999, pp. 78–89. Morgan Kaufmann Publishers Inc., San Francisco (1999)
88. Kim, C., Chhugani, J., Satish, N., Sedlar, E., Nguyen, A.D., Kaldewey, T., Lee, V.W., Brandt, S.A., Dubey, P.: FAST: Fast Architecture Sensitive Tree Search on Modern CPUs and GPUs. In: Proceedings of the 2010 International Conference on Management of Data, SIGMOD 2010, pp. 339–350. ACM, New York (2010)
 89. Volk, P.B., Habich, D., Lehner, W.: GPU-Based Speculative Query Processing for Database Operations. In: First International Workshop on Accelerating Data Management Systems Using Modern Processor and Storage Architectures (September 2010)
 90. Advanced Micro Devices, Inc.: AMD Demonstrates World’s First Fusion APU at Computex 2010 (December 2011), <http://www.amd.com/us/press-releases/Pages/amd-demonstrates-2010june02.aspx>
 91. Intel Corporation: Intel Atom Embedded Processors (December 2011), <http://www.intel.com/content/www/us/en/processors/atom/atom-processor.html>
 92. Advanced Micro Devices, Inc.: AMD Fusion Family of APUs: Enabling a Superior, Immersive PC Experience (December 2011), http://www.amd.com/us/Documents/48423_fusion_whitepaper_WEB.pdf
 93. Feldman, M.: First HPC Cluster with AMD Fusion Chips Debuts at Sandia (December 2011), <http://www.hpcwire.com/hpcwire/2011-11-02/first-hpc-cluster-with-amd-fusion-chips-debuts-at-sandia.html>
 94. Advanced Micro Devices, Inc.: Fusion for Servers (December 2011), <http://blogs.amd.com/work/2010/06/10/fusion-for-servers/>
 95. Intel Corporation: Intel Many Integrated Core Architecture (December 2011), <http://www.intel.com/content/www/us/en/architecture-and-technology/many-integrated-core/intel-many-integrated-core-architecture.html>
 96. Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T., Hanrahan, P.: Larrabee: A Many-core x86 Architecture for Visual Computing. In: ACM SIGGRAPH 2008 Papers, SIGGRAPH 2008, pp. 18:1–18:15. ACM, New York (2008)
 97. Intel Corporation: Teraflops Research Chip (December 2011), <http://techresearch.intel.com/ProjectDetails.aspx?Id=151>
 98. Intel Corporation: Single-Chip Cloud Computer (December 2011), <http://techresearch.intel.com/ProjectDetails.aspx?Id=1>
 99. Intel Corporation: The SCC Platform Overview (December 2011), http://techresearch.intel.com/spaw2/uploads/files/SCC_Platform_Overview.pdf
 100. Shilov, A.: Intel Shows Off “Knights Corner” MIC Compute Accelerator (December 2011), http://www.xbitlabs.com/news/cpu/display/20111115163857_Intel_Shows_Off_Knights_Corner_MIC_Compute_Accelerator.html
 101. Fang, W., He, B., Luo, Q., Govindaraju, N.K.: Mars: Accelerating MapReduce with Graphics Processors. IEEE Transactions on Parallel and Distributed Systems 22, 608–620 (2011)

An Introduction to Multicriteria Decision Aid: The PROMETHEE and GAIA Methods

Yves De Smet and Karim Lidouh

Computer and Decision Engineering Departement CoDE-SMG, Ecole polytechnique
de Bruxelles, Université libre de Bruxelles, Belgium

yves.de.smet@ulb.ac.be

<http://code.ulb.ac.be/~yvdesmet>

Abstract. Most strategic decision problems involve the evaluation of potential solutions according to multiple conflicting criteria. The aim of this chapter is to introduce some basic concepts of Multicriteria Decision Aid (MCDA) with a special emphasis on the PROMETHEE and GAIA methods. First, we will introduce the specific vocabulary of this research area as well as traditional modelling issues. The main part of the presentation will be dedicated to explain in detail the PROMETHEE and GAIA methods. Finally, an illustrative example will be analyzed with the D-Sight software. This will highlight the added value of using interactive and visual tools in complex decision processes.

Keywords: multicriteria decision aid, outranking methods, PROMETHEE, GAIA.

1 Introduction

Multicriteria Decision Aid (MCDA) has been an active field of research for more than 40 years. Summarizing it in a few pages is, of course, impossible. Consequently, the only ambition of this chapter is to constitute a rough introduction to the subject. Additionally, we have decided to detail a given methodology, namely the PROMETHEE and GAIA methods, rather than to present an oversimplified summary of different methods. As a consequence, the reader should keep in mind that plenty of other approaches do exist and deserve attention (for instance AHP [39], MAUT [20], ELECTRE [24], MACBETH [3], ...).

As shown hereafter, a brief analysis of the terms "*multicriteria decision aid*" already allows the novice to understand the underlying motivations of this research area [18]. It is, first of all, a decision **aid** activity (versus decision **making**) that has its root in the **multicriteria** paradigm. These statements will be further commented in the next two subsections. We refer the interested reader to [2,8,18,36,37,40,43,44] for detailed discussions.

1.1 What Is Decision Aid ?

Selecting an investment project, appointing a new employee, choosing a site to establish a garbage dump, diagnosing a disease, etc. All these examples show

that *deciding* is a complex activity that, in many cases, can have important consequences.

A decision is, first of all, the result of a more or less time consuming process that is made of partial decisions, negotiations and learning phases, search for (additional) information, etc. During this process, new potential solutions can appear while others become not feasible anymore. The context of the problem can be such that the evaluation of the potential solutions has to be made according to several conflicting points of views (possibly integrating subjective elements). The data are often imprecise, uncertain or simply not available. Social, economic and political constraints further increase the complexity of the decision process. Finally, most decisions involve several actors with different interests and goals.

Facing the complexity of this activity, one may try to *build* a model i.e. an abstraction of the reality that will be used, during the decision process, as a support for investigation and communication. The limited, approximate, and imperfect nature of this model has to remind us of its modesty. This observation has led Bernard Roy [37] to define decision aid as follows:

Definition 1. *Decision aiding is the activity of the person who, through the use of explicit but not necessary completely formalized models, helps obtain elements of responses to the questions posed by a stakeholder¹ in a decision process. These elements work towards clarifying the decision and usually towards recommending, or simply favoring, a behavior that will increase the consistency between the evolution of the process and this stakeholder's objectives and value system.*

1.2 What Is Multicriteria Decision Aid ?

In the fifties, the pioneers of Operational Research (O.R.) were convinced of the natural and promising applicability of their models. Twenty years later, the reality was somewhat different: some problems had been successfully treated by using classic operational research tools while, in other cases, their application had disappointed [35].

As noted by Schärliig [40], the success stories were essentially related to situations where the decision problem could have been *isolated* from its context: the search for optimal mixtures, an optimal traveling salesman problem, an optimal stock management, etc. In the other cases, the underlying assumptions of classic OR models appeared to be too restrictive to constitute an adequate model of the reality.

Indeed, most of unicriterion optimisation approaches rely on the following (implicit) assumptions [40]:

- stable set of actions: the set of alternatives is assumed to be known prior to the analysis and to remain unchanged during the decision process. On contrary, in most decision problems, new alternatives can appear during the analysis while others become not topical anymore.

¹ Here, the term stakeholder refers to any individual or entity that may intervene in the decision making process.

- exclusive actions: every alternative is assumed to perfectly reflect all the facets of the problem.
- transitivity: the preferences of the Decision Maker (DM) are assumed to be transitive. As a consequence it is possible to rank the alternatives from the worst to the best one and thus to find a so-called *optimal* solution.

Among the critics listed above, the one related to the non-transitivity of preferences is definitively the most crucial one. Indeed, in unicriterion optimisation models, one assumes that the decision maker is able to determine admissible alternatives i.e. those satisfying a given set of constraints. Then, these admissible alternatives are ranked according to the unique criterion (that is assumed to perfectly represent the preferences of the decision maker). Therefore it is possible to rank the alternatives from the worst to the best and to find a so-called “optimal” solution². As a consequence, in unicriterion optimisation models, the apparent universality of the *optimal solution* concept leads the analysts³ to search for a *hidden truth* [18,37,40,45].

”Where are you going on holidays next year?” This question has nothing to do with a complex optimisation or strategical decision problem. Yet it allows to illustrate the problem induced by multicriteria evaluations. Table 1 summarizes a fictitious problem. If you only consider the price, you should go hiking in the mountains. If you only consider to party, Ibiza is the best alternative. Obviously, there is no *objective* best ranking and therefore, no *objective* optimal solution (due to the conflicting nature of the criteria). If one agrees with these evaluations, the only objective information that could be stressed is that visiting the Pyramids in Egypt is a better choice than selecting a cruise in the Bahamas (since it is at least as good for all the criteria and strictly better for the price and culture). Then, you cannot compare the three other alternatives without adding subjective judgments such that *the criterion party is more important than culture, etc.*

As stressed in the previous example, the notion of *optimal* solution no longer exists in multicriteria contexts; researchers will rather look for *compromise solutions* i.e. solutions that are “*globally good*” according to the different criteria (without necessarily being the *best* for a given criterion) and that are not too bad on any given criterion.

We end this section by giving a few examples inspired by real applications. These will serve us through the chapter to illustrate the different concepts and methods.

Example 1. The Portfolio Management Problem (PMP). Let us consider a set of n equities and an investment capital K . The portfolio management

² In most cases, an alternative that is optimal for a specific criterion will not be optimal for another criterion (on the contrary, it is likely to be a bad solution according to this second point of view). In fact, most of people interpret the term *optimal* solution in an erroneous way because they assign it to a global meaning. On the contrary, in practice, one should ask the question “*optimal with respect to which criteria?*”

³ i.e. the person that helps the decision maker during the decision process.

Table 1. The holidays problem

| Type | Price | Exostism | Culture | Sports | Party |
|-------------------------|----------------|-----------|-----------|--------|-------------|
| Cruise in the Bahamas | Very expensive | Very good | Low | Low | Low |
| Pyramids in Egypt | Medium | Very good | Very good | Low | Low |
| Hiking in the mountains | Very low | Very low | Very low | High | Low |
| Party in Ibiza | Low | Medium | Very low | Good | Outstanding |

problem can be stated as follows: *”How much do we have to invest in the different equities in order to maximize the expected return and to minimize the risk?”* This famous question was first addressed by Markowitz [34]. Of course, there is not a unique portfolio that could be objectively considered as the best one (since the two criteria are in conflict: increasing the expected return will also increase the risk). In this problem, a crucial step is thus to compute the so-called Pareto-optimal frontier i.e. the set of portfolios such that the expected return cannot be increased without also increasing the risk (see Fig. 1; dots represent potential portfolios - the continuous curve represent the Pareto-optimal frontier). Once this set has been identified, the decision maker will have to select a given combination that best fit his risk aversion (or in other words his preferences). Let us stress that this bi-objective optimisation problem can easily be extended to the optimisation of other criteria such as liquidity, robustness, etc.

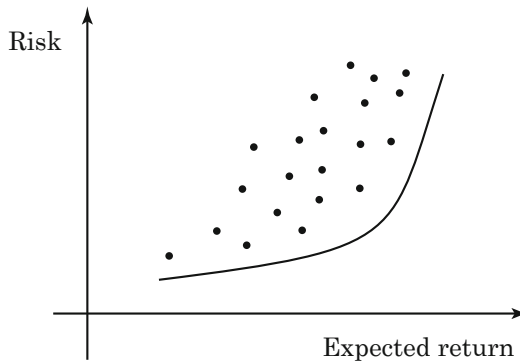


Fig. 1. Bi-objective portfolio problem

Example 2. The Criminality Assessment Problem (CAP). The Belgian police records statistics about criminal activities. These facts belong to predefined crime types such as robbery, road accidents, murders, burglary, sex offenses, prostitution, fraud, vandalism, etc. The severity of each crime type can be assessed according to different points of views: number of deaths, number of victims, financial impacts, social impacts, evolution over the last 5 years, type of organizations, etc. Every year, a ranking of these crime types is performed. This allows to allocate human, financial and material resources to efficiently fight the most

severe activities. Of course, modelling the severity of these crime types is not easy. If we consider the number of deaths, the worst crime type is related to road accidents. On the other hand, this is not related to criminal organizations and other related illegal activities such as in the case of prostitution. Vandalism is not characterized by a high number of victims or deaths and is not directly related to important criminal organizations however it has a high social impact, etc.

Example 3. The Diagnosis of Alzheimer’s Disease (DAD). Being able to detect Alzheimer’s disease is naturally of the uttermost importance. Any patient can be characterized by a set of criteria that are related to it. These encompass the age, heredity, etc. but also results to well-established memory tests. Given the evaluation of a patient, the problem consists of verifying if he or she suffers from this disease and assessing its severity. Once it is detected, one may consider three different grades: mild, medium, or severe.

Example 4. The Academic Ranking of World Universities (ARWU). Assessing the academic quality of world universities has become a topical issue over the last years. Based on criteria such as the number of articles published in top quality scientific journals, the number of awards received by alumni, etc. one establishes a ranking of institutions. If the so-called *Shanghai ranking* was initially developed to quantify the gap between Chinese and international universities, it is nowadays considered as a reference ranking that has been integrated in public life (see also section 3.3). Let us note that these rankings remain subject to criticism and has recently initiated a lot of debates [7].

2 Main Concepts and Terminology

Facing the complexity of a decision problem, the decision maker (DM) tries to rationalize it. Therefore, he has to identify the key elements that will intervene in the decision process i.e. the object of the decision, the set of potential solutions, a way to evaluate and compare them, the factors that can influence the decision(s), etc. This structuring phase is at the core of any multicriteria decision aid activity.

In this section, we will introduce the basic terminology that is used within the MCDA community and, consequently, increase the reader’s awareness of the MCDA problem’s formulation.

2.1 The Alternatives

At first, let us introduce the notion of an **action**. Intuitively, actions are the set of objects, alternatives, items, candidates, projects, potential decisions, etc. on which the decision is based. More formally,

Definition 2. [38] *An action is a generic term used to designate that which constitutes the object of the decision, or that which decision aiding is directed towards.*

In what follows, we will denote the set of actions $\mathcal{A} = \{a_1, \dots, a_n\}$. As stressed by Vincke [45], \mathcal{A} can be:

- *stable*: if \mathcal{A} can be defined a priori and is not likely to change during the decision process;
- *evolutive*: if, on the contrary, \mathcal{A} is likely to change during the decision process. Indeed, the decision process being an dynamic activity, intermediary results and/or the potential evolution of the decision context can lead to consider new actions while others are not topical anymore.

Furthermore, let us stress that \mathcal{A} is said to be **globalized**, if each element of \mathcal{A} excludes any others, and **fragmented** if it is not the case i.e. if combinations of elements from \mathcal{A} constitute possible outcomes of the decision process. Finally, one generally distinguishes contexts where \mathcal{A} can be defined by extension (the cardinality of \mathcal{A} is finite and relatively small. As a consequence, its elements can be enumerated) and situations where it is defined by comprehension (the cardinality of \mathcal{A} is infinite or relatively high. The elements of \mathcal{A} are identified as those satisfying a set of specific constraints).

In the Criminality Assessment Problem, the alternatives are the different pre-defined crime types. The set of alternatives is defined by extension (since its elements can be easily enumerated) and stable (unless a new form of crime type appears). In the Portfolio Management Problem, the set of alternatives is defined by comprehension i.e. all the investment options that do not exceed the capital limit.

2.2 The Criteria

Once the set \mathcal{A} has been determined, one has to characterize the actions (according to different points of views). This is formalized by the notion of criterion.

Definition 3. [45] *A criterion is a function f , defined on \mathcal{A} taking its value in a totally ordered set and representing the decision maker's preferences according to some point of views.*

$$f : \mathcal{A} \rightarrow E, \text{ where } E \text{ is a totally ordered set}$$

Without loss of generality, we will assume that all criteria have to be minimized⁴. Let $f_j(a_i)$ denote the evaluation of action a_i according to the criterion f_j . Let us assume that q distinct criteria are involved in the decision problem and let $F = \{f_1, \dots, f_q\}$ be the set of all criteria.

In the previous definition, we see that the only restriction about E is the fact that it is a totally ordered set. In other words, given two elements $e, e' \in E$ it is always possible to state if $e \succ e'$, $e = e'$ or $e' \succ e$. The poorest scale that respects this condition is the ordinal one. Of course, richer scales can be

⁴ We assume that any totally ordered set E can be represented by real numbers. If a given criterion has to be maximized, taking the symmetric values of the evaluations allows to consider it as a criterion to be minimized.

considered such interval or ratio scales (see [10]). These differ with respect to the kind of mathematical operations that are allowed. In the Diagnosis of Alzheimer's Disease, one may consider the judgment of a physician about the severity of the disease. Five values could be considered: very bad, bad, medium, good and very good. Even if these values are coded using respectively the numbers 0,1,2,3,4, one may only state that 1 is worst than 2 (bad is worst than medium). Saying that medium is two times better than bad is not correct (since this depends on the arbitrary nature of the coding). When modelling a multicriteria problem, the decision maker should always keep in mind the nature of the scale characterizing the different criteria since this will restrict the kind of mathematical operations that are allowed.

At this point of the analysis, the only objective information that can be extracted from the decision problem is based on the Pareto dominance relation:

Definition 4. Let D denote the Pareto dominance relation i.e. $aDb \Leftrightarrow f_j(a) \leq f_j(b) \forall j \in \{1, \dots, q\}$ and $\exists k \in \{1, \dots, q\} | f_k(a) < f_k(b)$.

This relation leads to distinguish efficient and dominated actions from \mathcal{A} .

Definition 5. An action a is said to be efficient if $\nexists b \in \mathcal{A} : bDa$

If the purpose of the Decision Maker is to select a single action, he would be tempted to first remove all dominated actions from \mathcal{A} . Unfortunately, the number of remaining efficient actions will still remain important (since generally there is no action that is simultaneously the best for all the criteria). On the other hand, one can explicitly build a *virtual* action, called observed ideal point, that satisfies the aforementioned condition:

Definition 6. The observed ideal point, $i(\mathcal{A})$, associated to \mathcal{A} , is the point the coordinates of which are $(i(\mathcal{A})^1, \dots, i(\mathcal{A})^q)$ where:

$$i(\mathcal{A})^j = \min_{a \in \mathcal{A}} f_j(a)$$

Since the ideal point (or assimilated actions i.e. that are the best for all criteria) does not usually belong to \mathcal{A} , the notion of optimal solution is not adapted to multicriteria problems. On the contrary, in most cases, the presence of conflicting criteria will lead the decision maker to rather focus on *compromise* solutions among efficient alternatives. As a consequence, he will be forced to express subjective judgements in order to make trade-offs between the different criteria, to interpret the evaluation scales, etc. Naturally, this leads to the question of formally modelling his preferences.

2.3 Preference Modelling

As already stressed, most multicriteria decision aid problems cannot be solved if we simply rely on the dominance relation (since the cardinality of the efficient set is too high). Therefore, additional information has to be asked to the decision

maker. This leads to the parametrization of a particular mathematical model to represent in the best possible way the choice of given decision maker. As a consequence, it is crucial to be able to represent his preferences in a formal way. This section will introduce the basics of preference modelling.

When modelling the decision maker’s preferences, one usually distinguishes the three following binary relations⁵: Preference (P), Indifference (I) and Incomparability (J), which result from the comparison between two actions a_i and $a_j \in \mathcal{A}$

$$\begin{cases} a_iPa_j & \text{if } a_i \text{ is preferred to } a_j \\ a_iIa_j & \text{if } a_i \text{ is indifferent to } a_j \\ a_iJa_j & \text{if } a_i \text{ is incomparable to } a_j \end{cases}$$

Indeed, these relations translate situations of preference, indifference and incomparability and it can be assumed that they satisfy the following requirements:

$$\forall a_i, a_j \in \mathcal{A} \begin{cases} a_iPa_j \implies a_i\neg Pa_j & : P \text{ is asymmetric} \\ a_iIa_i & : I \text{ is reflexive} \\ a_iIa_j \implies a_jIa_i & : I \text{ is symmetric} \\ a_i\neg Ja_i & : J \text{ is irreflexive} \\ a_iJa_j \implies a_jJa_i & : J \text{ is symmetric} \end{cases}$$

Definition 7. [45] *The three relations $\{P, I, J\}$ make up a preference structure on \mathcal{A} if they satisfy the above conditions and if, given any two elements a_i, a_j of \mathcal{A} , one and only one of the following properties is true: $a_iPa_j, a_jPa_i, a_iIa_j, a_iJa_j$.*

Intuitively [37]:

- aPb corresponds to the existence of clear and positive reasons that justify significant preference in favor of a;
- aIb corresponds to the existence of clear and positive reasons that justify equivalence between the two actions;
- aJb corresponds to an absence of clear and positive reasons that justify any of the two preceding relations.

In the classic *unicriterion* optimisation models, the pairwise comparisons of actions can only lead to two situations: preference or indifference. In the same way, many multicriteria methods, such as multi-attribute utility functions for instance, aggregate all the criteria into a unique (artificial) value and, therefore, transform the multicriteria problem into a unicriterion optimisation problem. In this context, both the indifference and preference relations are assumed to be transitive. These assumptions have, nevertheless, been criticized by several authors. For example, Luce [30] illustrates the non-transitivity of the indifference relation with the following example: let us consider 401 cups of coffee, noted C_0, C_1, \dots, C_{400} . One assumes that the cup C_i contains exactly $(1 + \frac{i}{100})$ grams of sugar. In this context, any normal person is unable to differentiate two successive cups. Therefore, we have: $C_0IC_1, C_1IC_2, C_2IC_3, \dots, C_{399}IC_{400}$. However, it is obvious that nobody will state C_0IC_{400} .

⁵ R is a binary relation on $\mathcal{A} \Leftrightarrow R \subseteq \{(a_i, a_j) | a_i, a_j \in \mathcal{A}\}$

Let us note that some authors [37] further enrich the previous structure by a relation Q which stands for a *weak preference* relation (versus the *strict preference* relation P). In other words, if $a_i Q a_j$, the decision maker knows that $a_j \neg P a_i$ but cannot clearly choose between $a_i I a_j$ or $a_i P a_j$. However this specific relation will not be considered in the present work.

The potential presence of incomparability is a distinctive feature of the so-called French school of multicriteria decision aid. As already stressed, $a J b$ is stated when the decision maker cannot clearly choose among the three possibilities: $a P b$, $b P a$ or $a I b$. This can happen, for instance, due to a lack of information, to uncertainty or conflicting preferences (see [37] for illustrative examples).

Finally, let us define the relation $\mathcal{S} = (P \cup I)$. Thus, $a_i \mathcal{S} a_j$ will stand for a_i is at least as good as a_j . A direct consequence of this definition is:

$$\forall a_i, a_j \in A \begin{cases} a_i P a_j \Leftrightarrow a_i \mathcal{S} a_j, a_j \neg \mathcal{S} a_i \\ a_i I a_j \Leftrightarrow a_i \mathcal{S} a_j, a_j \mathcal{S} a_i \\ a_i J a_j \Leftrightarrow a_i \neg \mathcal{S} a_j, a_j \neg \mathcal{S} a_i \end{cases}$$

We refer the interested reader to [9] for a detailed introduction to binary relations and preference modelling.

Until now, we have restricted ourselves to binary relations for preference modelling. Let us note that another important trend relies on valued relations. This will be illustrated in section 3.1 which presents the PROMETHEE methods.

2.4 Consistent Family of Criteria

A fundamental difficulty in multicriteria decision aid is to represent the decision maker’s preferences on the basis of the evaluations of the actions according to the different criteria. The selection of these criteria is thus a crucial first step of the modelling activity. One way to formalize this selection is to require certain properties such as exhaustivity, cohesion and non redundancy. Intuitively:

- **exhaustivity**: if a_i and a_j are two actions that are identical with respect to all criteria, then one cannot have $a_i P a_j$, $a_j P a_i$ or $a_i J a_j$. Should one of these relations hold, then at least one other differentiating criterion would have been forgotten and would thus ought to be added to the set of considered criteria.
- **cohesion**: let us assume that a_i and a_j are indifferent ($a_i I a_j$). Weakening a_i and reinforcing a_j on one criterion (different or the same) lead to $a_i (P \cup I) a_j$. This condition ensures some coherence between the criteria and the global preferences.
- **non redundancy**: the family of criteria $F = \{f_1, f_2, \dots, f_q\}$ does not contain any redundant criteria in the sense that the family obtained by removing any single criterion f_j from F would violate at least one of the two previous conditions.

These three properties put together allows to define a **consistent family of criteria**. We refer the interested reader to [5,26,29,37] for formal definitions.

2.5 The Different Multicriteria Problematics and Methods

Now that the basic multicriteria terminology and notions have been introduced, we are ready to define a **multicriteria decision problem** is.

Definition 8. [45] *A multicriteria decision problem can be defined as a situation where given a set of actions \mathcal{A} and a consistent family of criteria F over \mathcal{A} , we want to solve one of the following problems:*

- *determine a subset of actions considered as the best considering F (choice problem),*
- *partition \mathcal{A} in subsets with respect to pre-established norms (sorting problem), or*
- *rank order the set of actions \mathcal{A} from best to worst (ranking problem).*

Of course, many real problems involve a mixture of these three main issues. Moreover, additional considerations may be cited:

- **The description problem:** helps to describe actions and their consequences in a formalized and systematic manner to develop a cognitive procedure [37].
- **Choosing k among m actions** [2]: this problematic can be viewed as a mixed of choice and ranking problematics.
- **The design problem:** to search for, identify or create new decision alternatives to meet the goals and aspirations revealed through the MCDA process [6].
- **The portfolio problem:** to choose a subset of alternatives from a larger set of possibilities, taking into account not only the characteristics of the individual alternatives, but also the manner in which they interact (their positive and negatives synergies [6]).
- **The clustering problem:** to define homogeneous groups of alternatives with respect to the preferences of the decision maker. These can be ordered (see for instance [21]) or nominal (see for instance [19]).

Of course, the different problematics allow to clearly identify the final goal of the decision. Obviously, in the Portfolio Management Problem, we are facing a choice problematic since we are looking to select a given portfolio from the Pareto Optimal frontier (α problematic). By definition, in the Criminality Assessment Problem, we are trying to rank the different crime types (β problematic). In the Diagnosis of Alzheimer's Disease, we are sorting a given patient into one of the following categories: healthy, mild, medium, severe (γ problematic).

Different methods have been developed in order to address these problematics. Without being exhaustive, we can distinguish three main families [45]:

- **Interactive methods:** these techniques are based on strong interactions with the Decision Maker. After a first computation step, an initial solution is proposed. If the current solution is not satisfying, the DM reacts by providing additional information about his preferences (for instance; "I would

- like to improve the value of the current solution on a specific criterion and, therefore, I do accept to deteriorate it on other criteria”). This information is integrated in the optimization model and a new solution is computed. The process is repeated until it converges towards a satisfying solution (see [28]);
- **Multiple attribute utility theory:** these methods rely on the assumption that all criteria can be aggregated into a single function that has to be optimized. Therefore, the multicriteria problem is transformed into a single optimization problem (see for instance UTA [41], AHP [39], MACBETH [3], etc.);
 - **Outranking methods:** these approaches are based on the construction and the exploitation of an outranking relation [45]:

Definition 9. *An outranking relation is a binary relation S defined in A such that aSb if, given what is known about the decision-maker’s preferences and given the quality of the evaluation of the actions and the nature of the problem, there are enough arguments to decide that a is at least as good as b , while there is no essential reason to refute the statement (Bernard Roy).*

Major families of outranking methods are ELECTRE [24] and PROMETHEE [17].

3 The PROMETHEE and GAIA Methods

3.1 PROMETHEE

PROMETHEE [6] I (partial ranking) and PROMETHEE II (complete ranking) were developed by J.P. Brans and presented for the first time in 1982 at a conference organized by R. Nadeau and M. Landry at the Université Laval, Québec, Canada (L’Ingénierie de la Décision. Elaboration d’Instruments d’Aide à la Décision). Since this seminal work, a lot of developments [11,13,16,17] have been proposed including visual representations [33], tools for robustness and sensitivity analysis [14,32], an extension to address the portfolio problematic, called PROMETHEE V, etc. More recently, a literature review [4] listed more than 200 PROMETHEE-based papers published in 100 different journals. The application fields cover finance, health care, logistics and transportation, hydrology and water management, manufacturing and assembly, etc.

The PROMETHEE methods are based on pairwise comparisons. When comparing two actions a_i and a_j on criterion f_k , the difference of evaluations between these two actions should be taken into account. Assuming that criterion f_k has to be minimized, this difference can be stated as follows,

$$d_k(a_i, a_j) = f_k(a_j) - f_k(a_i)$$

⁶ PROMETHEE is the acronym of Preference Ranking Organisation METHod for Enrichment Evaluations.

When the difference $d_k(a_i, a_j)$ is small and the DM can neglect it, there is no reason to say that a_i is preferred to a_j and consequently the actions are indifferent (for the specific criterion f_k). The higher the value of d_k , the larger the preference $P_k(a_i, a_j)$ in favor of a_i over a_j , on criterion f_k . This preference can be defined through a *preference function* in the following way,

$$P_k(a_i, a_j) = H_k(d_k(a_i, a_j)), \quad \forall a_i, a_j \in \mathcal{A}$$

and we can assume that $P_k(a_i, a_j) \in [0, 1]$ (if $P_k(a_i, a_j) > 0$, then $P_k(a_j, a_i) = 0$).

The pair $(f_k, P_k(a_i, a_j))$ is called a *generalized criterion* associated with criterion f_k , for all $k \in \{1, \dots, q\}$. Generally, 6 types of generalized criteria are considered. Generalized criterion of type 5 requires the definition of both q_k and p_k (see Fig. 2). The value p_k is called the preference threshold and is defined as the smallest difference on criterion f_k between actions for which the decision maker can say without a doubt that he prefers the better one. Similarly, q_k is called the indifference threshold and is defined as the biggest difference on criterion f_k for which the decision maker can say without a doubt that he is indifferent between the two actions.

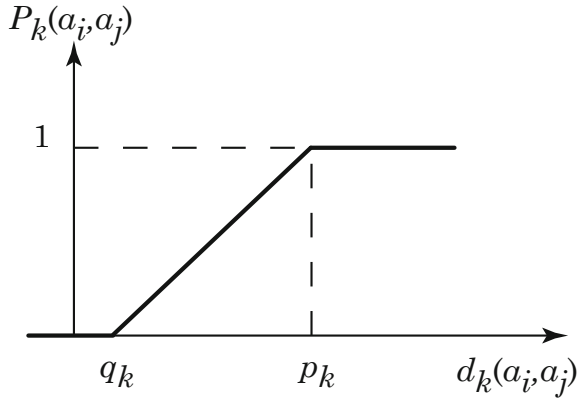


Fig. 2. Generalized criterion of type 5

Once the preference degrees between two actions a_i and a_j have been computed for every criterion, one needs to aggregate this marginal contribution to obtain $P(a_i, a_j)$ i.e. a global measure of the preference of a_i over a_j . This is performed using a classical weighted sum (ω_k is assumed to be the weight associated to criterion f_k):

$$P(a_i, a_j) = \sum_{k=1}^q \omega_k \cdot P_k(a_i, a_j)$$

$P(a_i, a_j)$ represents the valued preference of a_i over a_j (and not a binary preference as introduced in Section 2.3). Obviously, we have

$$P(a_i, a_j) \geq 0$$

and

$$P(a_i, a_j) + P(a_j, a_i) \leq 1.$$

The fundamental idea underlying the PROMETHEE methods is the quantification of how an action a outranks all the remaining $(n - 1)$ actions and how a is outranked by the other $(n - 1)$ actions. This idea leads to the definition of the positive $\phi^+(a)$ and negative $\phi^-(a)$ outranking flows. More formally:

$$\phi^+(a_i) = \frac{1}{n - 1} \sum_{a_j \in \mathcal{A}, i \neq j} P(a_i, a_j)$$

$$\phi^-(a_i) = \frac{1}{n - 1} \sum_{a_j \in \mathcal{A}, i \neq j} P(a_j, a_i)$$

Given these two measures, two total pre-orders⁷ of \mathcal{A} can be obtained (one associated to the values of ϕ^+ and another associated to the values of ϕ^-). The intersection of these two pre-orders leads to a partial pre-order called the PROMETHEE I ranking. In this context, two actions a_i and a_j will be judged to be incomparable if $\phi^+(a_i) > \phi^+(a_j)$ and $\phi^-(a_i) > \phi^-(a_j)$ or if $\phi^+(a_i) < \phi^+(a_j)$ and $\phi^-(a_i) < \phi^-(a_j)$.

On the other hand, the complete pre-order obtained with the PROMETHEE II method is based on the net flow $\phi(a_i)$ assigned to each action $a_i \in \mathcal{A}$.

$$\phi(a_i) = \phi^+(a_i) - \phi^-(a_j)$$

Let us note that,

$$\phi(a_i) = \frac{1}{n - 1} \sum_{k=1}^q \sum_{a_j \in \mathcal{A}} \{P_k(a_i, a_j) - P_k(a_j, a_i)\} \cdot \omega_k = \sum_{k=1}^q \phi_k(a_i) \cdot \omega_k$$

where $\phi_k(a_i)$ is called the k^{th} unicriterion net flow assigned to action a_i . Intuitively, these values allow to better position action a_i , according to criterion f_k , with respect to all the other actions in \mathcal{A} .

In addition to these rankings, Mareschal and Brans [33] have proposed a geometrical tool that helps the decision maker both to interactively explore and structure the decision problem, and to better understand the results provided by the PROMETHEE rankings. This is referred to as the GAIA plane. The underlying idea of this approach is to perform a principal components analysis on the unicriterion net flows assigned to each action [16]. This will be further developed in the next section.

⁷ A pre-order is a binary relation that is both transitive and reflexive.

3.2 GAIA and Its Interpretation

When we take our set of alternatives into account, it is often difficult to get a visual representation of it because of the numerous criteria that we try to keep in mind. Indeed, if we think of a multidimensional space defined by taking each of those criteria into account, the alternatives could be represented as points that each have specific coordinates depending on their evaluations. Such a space is represented in Fig. 3 with a set of actions positioned with respect to five criteria. Of course, it is impossible to represent a five dimensional space on paper and the representation in Fig. 3 is merely a projection of the actual space on a two dimensional plane. Furthermore, if the view point of such a projection is poorly chosen, the representation will rarely teach us anything useful.

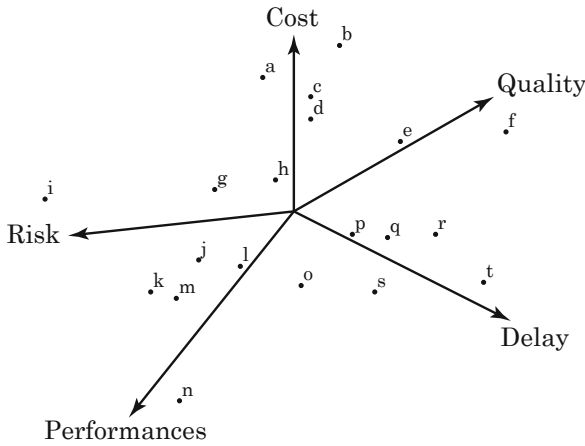


Fig. 3. Criteria space

The aim of GAIA⁸ will be to find the best view point for this multidimensional space in order to extract as much information as possible from the representation. In order to do that we are going to resort to a Principal Component Analysis (PCA) applied on the unicriterion net flows $\phi_k(a_i)$ computed by PROMETHEE. This will ensure that the actions we represent will be defined as they are seen by the decision maker. Indeed, the unicriterion net flows contain information on how the decision maker perceives the different criteria and compares the actions pair by pair.

Let us consider a matrix Φ that contains all the unicriterion net flows for our problem. We have:

$$\Phi = (\phi_k(a_i)) \quad \forall a_i \in \mathcal{A}; k \in \{1, 2, \dots, q\}$$

We begin by calculating the variance-covariance matrix C of our problem.

$$nC = \Phi' \Phi$$

⁸ GAIA is the acronym of Graphical Analysis for Interactive Assistance.

We then compute the eigenvectors and eigenvalues of matrix C . All of these eigenvectors are orthogonal because of the properties of matrix C and they each indicate a direction towards which we have a certain dispersion of the alternatives' positioning. That dispersion is given to us by the respective eigenvalues of each vector.

Finally, we select the two eigenvectors u and v with the highest associated eigenvalues λ_1 and λ_2 and use those to define a two-dimensional plane in the criteria space. This plane will be the canvas that will be used to represent the decision problem and all of its defined elements (i.e. the actions, the criteria, the direction of the best compromise).

Since we have selected the vectors with the highest eigenvalues to define our plane, it means the plane will capture the maximum dispersion of the alternatives in two dimensions. It is also possible to evaluate the amount of information kept that way. That amount is called the delta value of the plane and is denoted δ :

$$\delta = \frac{\lambda_1 + \lambda_2}{\sum_{k=1}^q \lambda_k}$$

Once the plane for the projection has been defined, we project the actions defined by their coordinates (i.e. the unicriterion net flows) on it. The actions' coordinates in the criteria space can be written as:

$$\alpha_i : (\phi_1(a_i), \phi_2(a_i), \dots, \phi_k(a_i), \dots, \phi_q(a_i)), \forall a_i \in \mathcal{A}$$

The projection of the actions can thus be found as follows:

$$\begin{cases} |Op_i| = \alpha'_i u \\ |Oq_i| = \alpha'_i v. \end{cases}$$

We then add the projection of the axes e_k representing each criterion on the plane:

$$e_k : (0, 0, \dots, 1, 0, \dots, 0) \quad k \in \{1, 2, \dots, q\}.$$

Those projections are denoted c_k .

Finally, to give an idea of which actions are closest to the best compromise, we add the projection of the weights vector.

$$w : (w_1, w_2, \dots, w_k, \dots, w_q).$$

That projection is often referred to as the decision stick and is computed as follows:

$$\pi = \sum_{k=1}^q c_k \cdot \overline{w_k},$$

where $\overline{w_k}$ is the k -th coordinate of the normalised vector corresponding to w i.e. $\overline{w} = w/|w|$.

When all of the components have been added, we are able to display a projection similar to the one on Fig. 4. All of these elements and their relative positions can now be interpreted.

$$\delta = 78\%$$

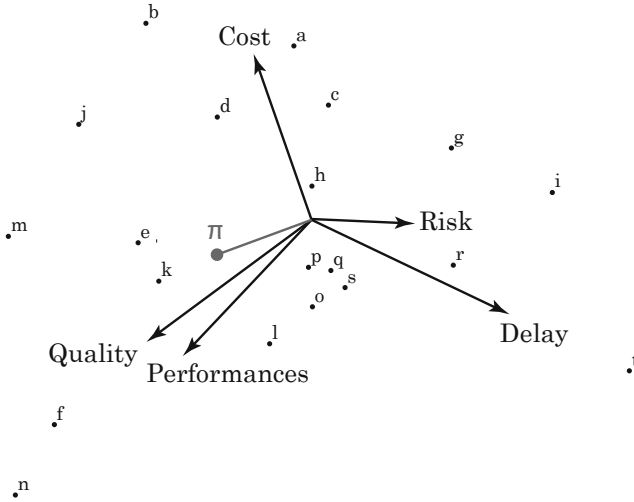


Fig. 4. GAIA plane

- **Positions of the criteria:** The orientation of the axes will indicate which criteria are compatible and which ones are in conflict. We can see for example that in the case of Fig. 4 the axes for “Quality” and “Performances” are very close to each other and therefore compatible. That means that we can easily find alternatives that excel in both quality and performance simultaneously, or, on the contrary, that some alternatives have bad evaluations on both of these criteria. Also, we can notice that “Cost” and “Delay” are in conflict because they are pointing towards very different directions. The same can be said for “Quality” and “Risk”. This means that it is very difficult to find an action that presents good scores on both criteria for each pair. Usually, when an action is good on one of those criteria, it is bad on the other.

Furthermore, the size of the criteria axes will point out the discriminant criteria within the problem. Indeed, since the plane has been chosen to capture the maximum variation of the actions, the criteria that do not present a high enough variation of the evaluations will likely end up being orthogonal to the plane. In this last case we can see that all the criteria axes have a relatively good size with the exception of “Risk”. We can therefore say that the risk measured on the actions for this problem does not differentiate them as well as the other criteria do.

- **Position of the decision stick π :** In this multivariate view, the indication of an objective is of high importance. It will indicate us the importance that the decision maker has given to each criterion. In the representation on Fig. 4 the decision stick points slightly more towards “Quality” and

“Performances” than the other criteria. The weight associated to those two criteria must therefore be bigger than for criteria such as “Risk” or “Delay”. Of course, changing the weights i.e. the relative importances of the criteria, will change the direction of the decision stick and make it point in a different direction.

- **Relative positions of the alternatives:** Groups of alternatives on the plane will represent solutions with similar profiles. Actions o , p , q , and s seem to have the same characteristics. But also, alternatives b , n , and t have very different profiles that ultimately give them projections very far from each other.
- **Positions of the alternatives (according to the criteria):** The location of an alternative on the plane will give us an indication on the type of profile it has. It will point out the strongest and weakest features of a solution. By taking a look at the actions in the direction of each criterion, we can see that actions n and f have strong evaluation in quality and performances but low ones on risk. Action t has a good evaluation on delay and a fairly good one regarding risk. Actions a , b , c , and d are oriented towards costs and behave poorly in terms of delay. Actions m and j are good on cost, quality, and performance, but bad on delay and risk.
- **Positions of the alternatives (according to the decision stick):** When projected on the decision stick, the alternatives take their positions from the PROMETHEE II ranking. Even though the ranking inferred from a projection could present differences due to loss of data, it still is an interesting use of the tool when more precise information is not available. In our example, the inferred ranking we obtain would be, from best to worst: n , f , m , e , k , j , b , l , d , o , p , q , s , h , a , c , r , g , i , t .
- **Delta value δ :** These results would not be complete without an indication on their reliability. The delta value i.e. the amount of information preserved by the plane, will give us a confidence level for the results and will have to be indicated alongside them. In most software implementations, the delta value is therefore given in one of the corners of the plane as a percentage. In the given representation a value of 78% means that 22% of the variation of the actions is lost and not represented on the plane. We can thus say that the two dimensions that were chosen for this projection due to their associated eigenvalues successfully represent 78% of the information from the five criteria in this problem.

The results we extract from the GAIA plane are, of course, an approximation of the reality. Because of the loss of data due to the projection on the plane, some of the actions might not be well represented in two dimensions. For example, alternatives that seem close on the plane, might actually be apart from each other but have projections that are close. Every time we use the GAIA plane to draw conclusions, we will need to pay attention to the delta value and compare the inferred ranking to the complete ranking from PROMETHEE II.

3.3 A Pedagogical Example with D-Sight

During the recent years, we have witnessed the development of indexes allowing to evaluate countries, cities, universities, companies, etc. Among them, we may cite the Human Development Index (HDI), the Environmental Performance Index (EPI), the Global Peace Index (GPI), the Academic Ranking of World Universities (ARWU), the European Cities Monitor, etc. These evaluations are, of course, typical multicriteria decision aid problems that are, most of the time, solved by using a classical weighted sum (after a first normalization step). In the end, all the alternatives are characterized by a global score allowing to rank them from the best to the worst one. These results are often easily available on the web.

In what follows, we do not claim that computing a net flow score (like in the PROMETHEE II ranking) instead of a weighted value is more appropriate. We let this methodological question to further investigations. However, we assert that "*solving*" a multicriteria decision aid problem cannot be limited to the computation of a global score in order to rank the alternatives. In what follows, we will illustrate different steps that could lead to a better understanding of the problem and to more robust conclusions. In order to illustrate this, we will consider the ten first ranked universities of the ARWU in the field of computer sciences (see table 2): Stanford University (SU), Massachusetts Institute of Technology (MIT), University of California Berkeley (UCB), Princeton University (PU), Carnegie Mellon University (CMU), Cornell University (CU), University of Southern California (USC), The University of Texas at Austin (UTA), Harvard University (HU) and University of Toronto (UT). The universities are evaluated according to 5 criteria 11 (their relative importance is given between the parentheses):

- Alumni (10%): number of alumni from the institution winning Turing Awards in Computer Science since 1951;
- Awards(15%): staff of an institution winning Turing Awards in Computer Science since 1961;
- HiCi (25%): highly cited researchers in Computer Science category;
- PUB (25%): papers Indexes in Science Citation Index-Expanded in Computer Science;
- TOP (25%): percentage of papers published in the top 20% journals on the field of Computer Science compared to the papers published in all journals of that subject field;

We refer the interested reader to 11 for a detailed description of these criteria and their computation. Of course, we are aware of the fact that such kind of rankings are subject to criticisms. However, this debate exceeds the illustrative purpose of this section.

We propose to use the D-Sight software 25, that implements the PROMETHEE and GAIA methods, in order to analyze the problem. For the sake of simplicity, we have decided to use linear preference functions (with an indifference threshold equal to 0 and a preference threshold equal to 100) for all the

Table 2. Evaluations of the 10 first Universities listed in the ARWU in Computer Sciences for 2010

| Name | ARWU score | Alumni | Awards | HiCi | PUB | TOP |
|------|------------|--------|--------|------|------|------|
| SU | 100 | 90,7 | 86,6 | 100 | 80,9 | 97,9 |
| MIT | 94,8 | 54,2 | 100 | 89,2 | 87,8 | 89,3 |
| UCB | 82,7 | 100 | 96,8 | 42,9 | 76,7 | 86,1 |
| PU | 78,7 | 68,6 | 71,8 | 60,6 | 63 | 94,7 |
| CMU | 76,4 | 42 | 79,1 | 55,3 | 85,4 | 75,4 |
| CU | 67,9 | 42 | 57,3 | 55,3 | 57,3 | 85,5 |
| USC | 66,6 | 0 | 39,5 | 65,5 | 68,4 | 86,8 |
| UTA | 66,3 | 42 | 39,5 | 55,3 | 70,4 | 77,2 |
| HU | 65,6 | 97 | 0 | 42,9 | 65,5 | 93,7 |
| UT | 65,5 | 24,3 | 53 | 49,5 | 71,1 | 78,3 |

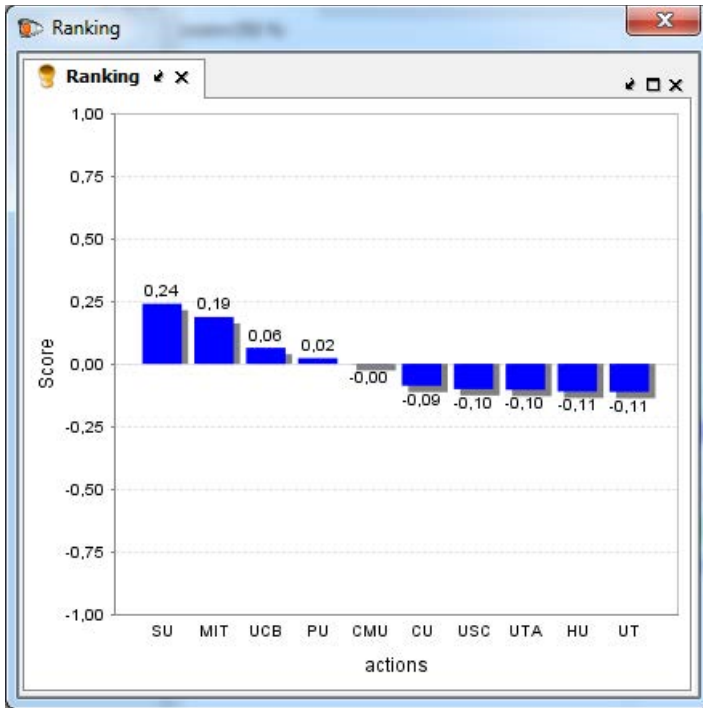


Fig. 5. Promethee II ranking

criteria (one more time, the aim of this section is to demonstrate the usefulness of visual and interactive tools in MCDA rather than to justify modelling choices). Additionally, this parametrization leads to the same ranking as the one induced by the ARWU score (see Fig. 5). As already stressed, in most cases, the analysis is stopped at this level i.e. the ranking of the alternatives according to

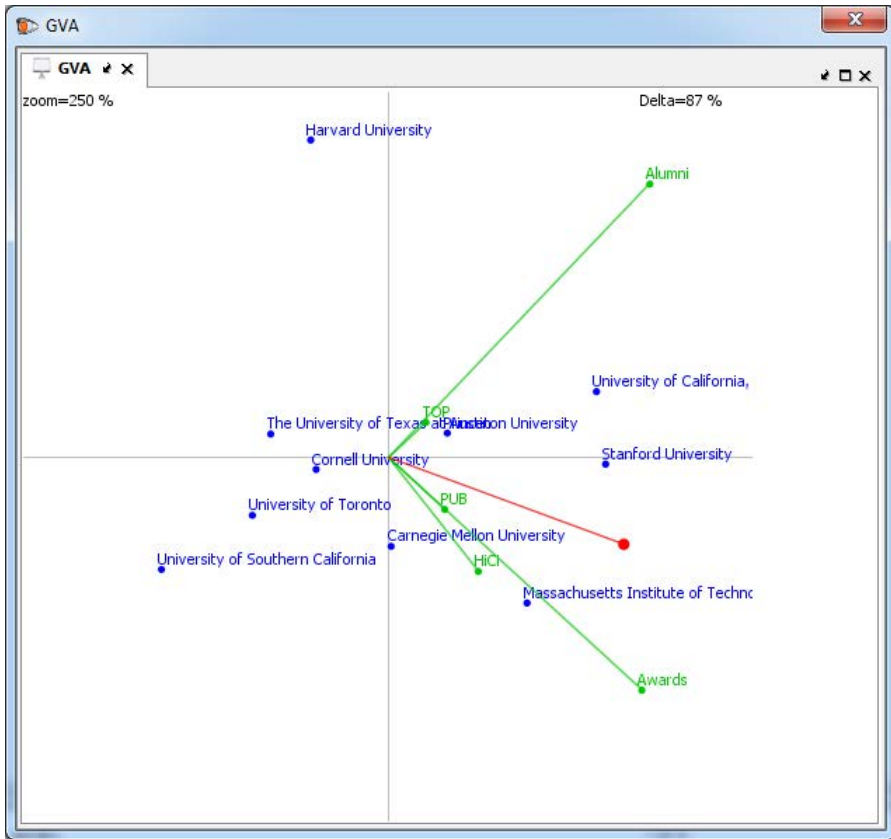


Fig. 6. GAIA plane

their scores. Let us now investigate how a software like D-Sight can help us to deepen our understanding of the problem.

A look on the GAIA plane (see Fig. 6) already helps us drawing some interesting conclusions:

- **Delta value:** the delta value is equal to 87%, which is a rather important value; the information loss due to the projections seems to be limited;
- **Relative positions of the criteria:** two groups of criteria can be identified: $\{PUB, HiCI, Awards\}$ and $\{Alumni, TOP\}$. These two sets seem to be independent from each other. In other words, there are no strong conflicts between the criteria;
- **Relative positions of the alternatives:** clearly the Harvard University is distinguishing itself from the cloud of other universities. Additionally, one may observe a similar effect for the group constituted by the University of California and Stanford University: these two institutions seem to

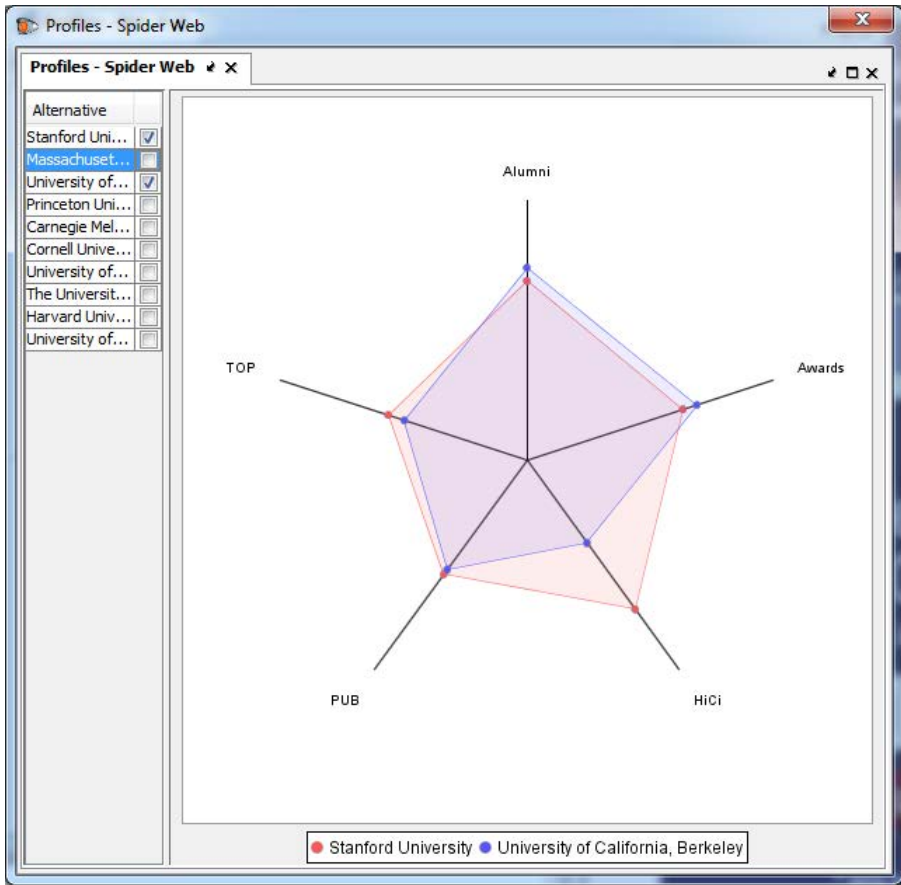


Fig. 7. Spider chart between SU and UCB

have close profiles. This is indeed confirmed by the spider chart shown on Fig. 7.

- **Relative position of the alternatives with respect to the criteria:** the Harvard University has a very particular evaluation; it is very good regarding Alumni and TOP and bad or average for the other criteria. Clearly, the University of California and the Stanford University have average good scores for both families of criteria. The Massachusetts Institute of Technology, which is ranked at the second position in the PROMETHEE II ranking, is very good on all criteria but has an average score on the Alumni criterion;
- **Decision Stick:** the projection of the different alternatives on the decision stick allows to find the total ranking (especially for the first ranked alternatives, see Fig. 8).

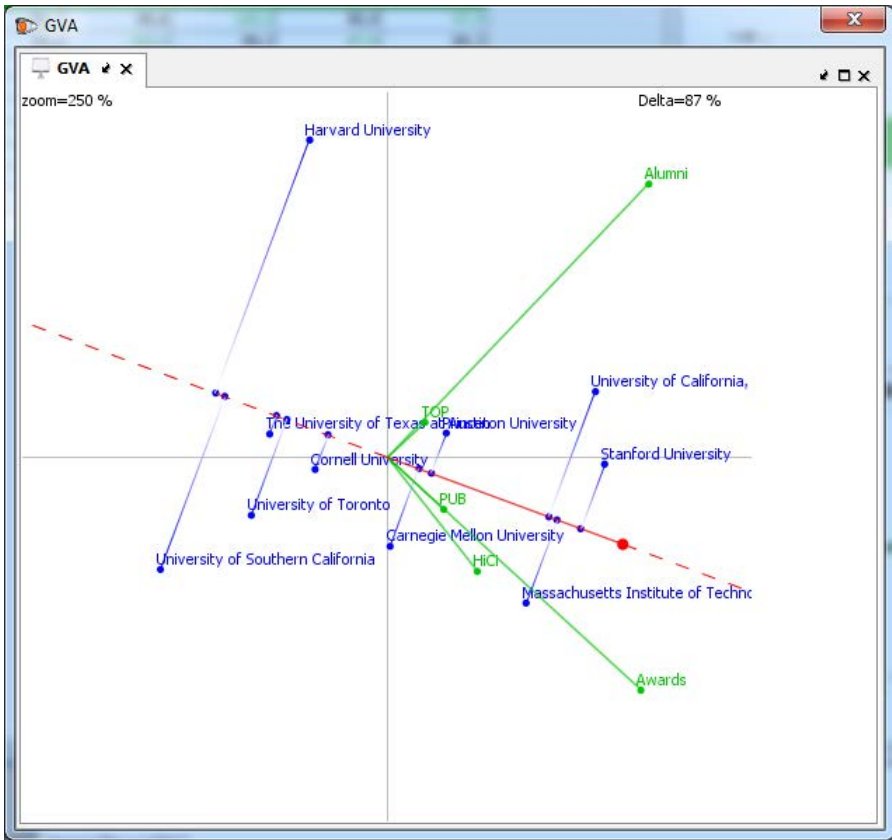


Fig. 8. Projection on the Decision Stick

As already stressed, a number of authors have criticized the legitimacy of such kind of rankings. Among others, the weight values can be discussed. If we slightly change the relative importance of a given criterion, would we get a totally different ranking? An interactive tool called *walking weights* allows the decision maker to perform a sensitivity analysis directly on the results while changing the weight values. For instance, multiplying the relative importance of the Alumni criterion by three (while the relative importance of other criteria remains the same) does not have an impact on the first ranked alternative (see Fig. 9). Finally, one could also address this question in a different way: *For every criterion, what are the interval values that will not affect the first or the two first ranked alternatives (under the assumption that the relative importance of other criteria remain the same)?* Fig. 10 shows these values when we want to hold the top ranking constituted by the two first alternatives. Clearly, we may observe that the interval values are rather large. More particularly, we may notice that even important modifications of the weight values of HiCi, PUB and TOP will not affect the top of the ranking. This proves that the selection of the two first alternatives seems to be rather robust.

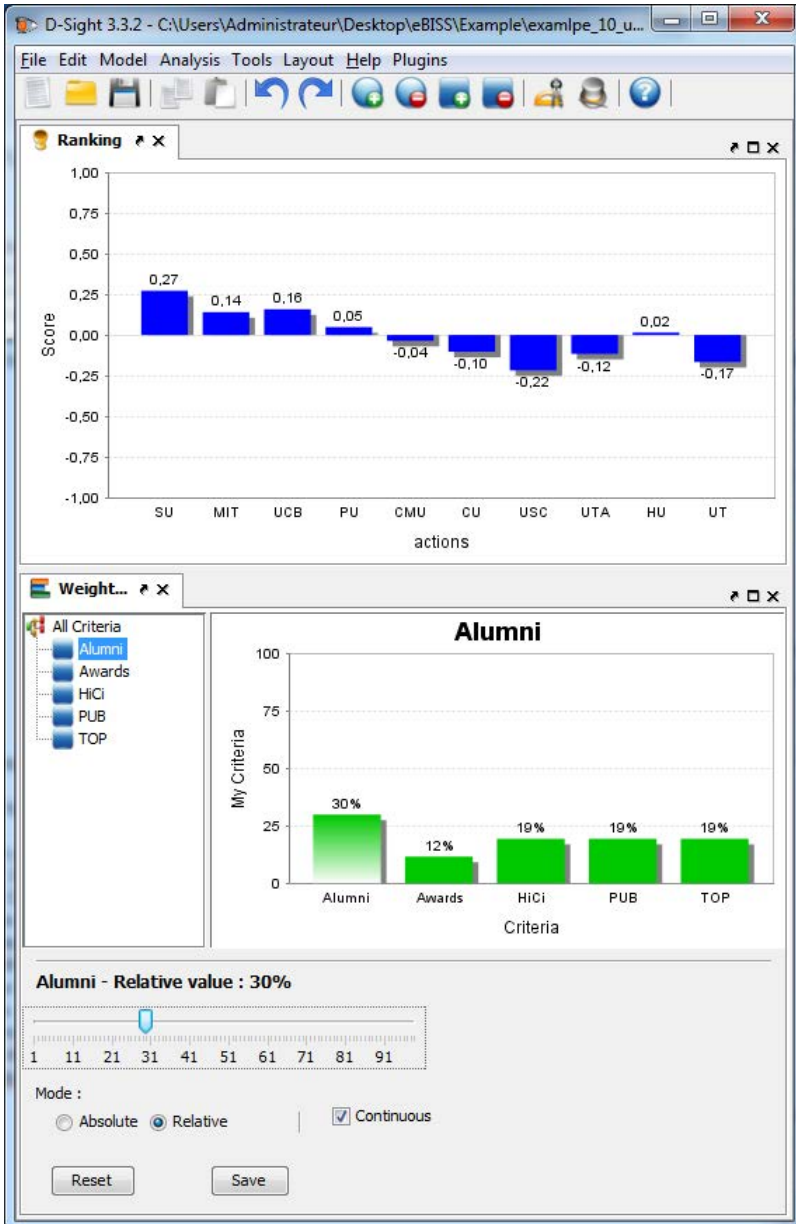


Fig. 9. Walking Weights

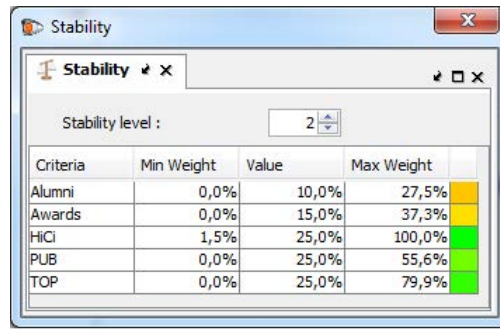


Fig. 10. Stability Intervals

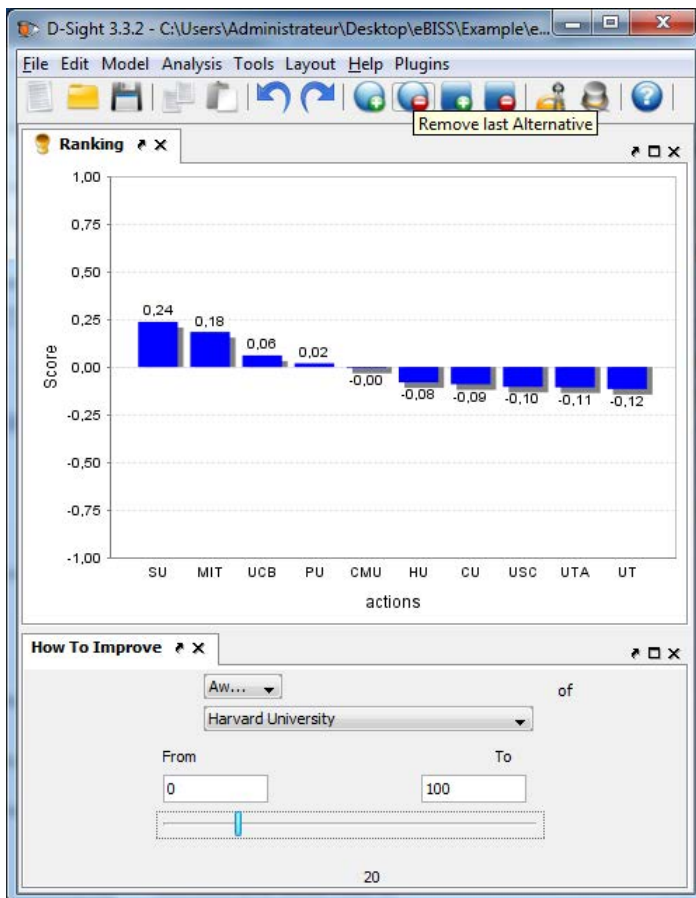


Fig. 11. How to improve?

Finally, a last strategical question could be: *How does the Harvard University need to improve itself on a given criterion in order to gain some positions?* Fig. 11 shows that increasing the score of the criterion Awards to 20 (which remains a relatively low value with respect to the evaluation table) allows Harvard to pass from the 9th position to the 6th.

This section has shown that the success of a multicriteria analysis heavily depends on the availability of user-friendly software. D-Sight is the third generation of PROMETHEE-based software (following PROMCALC [13] and Decision Lab 2000). We may not conclude this section without citing other multicriteria software such as Expert Choice (for the AHP method), Electre IS (for a generalization of the ELECTRE I method), M-Macbeth (as expected for the Macbeth method), etc. Another interesting initiative that has to be mentioned is the Decision Deck project, which is an open source software that is collaboratively developed and which implements various methods. The reader is also referred to [46] for a review on MCDA software.

4 Conclusion

Multicriteria decision aid is an exciting research field. The only ambition of this chapter was to introduce the basics of this domain. We refer the interested reader to [23] for a recent and complete state of the art of MCDA. Additionally, interesting resources can be found on the websites of the EURO working group on MCDA or of the MCDM international society.

As a demonstration of the growing interest in MCDA, we may point out that it has been applied to other research areas such as Artificial Intelligence [22], Geographic Information Systems [31], Classification and Pattern Recognition [19,21], System Dynamics [15], Group Decision and Negotiation [27], Scheduling [42], etc.

References

1. Academic Ranking of World Universities, <http://www.arwu.org>
2. Bana e Costa, C.A.: Convictions et Aide à la Décision. Newsletter of the Euro Working Group Multicriteria Decision Aiding 2 (1993)
3. Bana e Costa, C.A., De Corte, J.M., Vansnick, J.C.: On the Mathematical Foundations of MACBETH. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) Multiple Criteria Decision Analysis: State of the Art Surveys, pp. 133–162. Springer, Boston (2005)
4. Behzadian, M., Kazemzadeh, R.K., Albadvi, A., Aghdasi, M.: PROMETHEE: A comprehensive literature review on methodologies and applications. European Journal of Operational Research 100(1), 198–215 (2010)
5. Belton, V., Ackerman, F., Shepherd, I.: Integrated support from problem structuring through to alternative evaluation using COPE and VISA. Journal of Multi-Criteria Decision Analysis 7, 115–130 (1997)
6. Belton, V., Stewart, T.: Multiple Criteria Decision Analysis: An Integrated Approach. Kluwer Academic Publisher, Boston (2002)

7. Billaut, J.C., Bouyssou, D., Vincke, P.: Should you believe in the Shanghai ranking? *Scientometrics* 84(1), 237–263 (2010)
8. Bouyssou, D.: *Décision Multicritère ou Aide Multicritère?* Newsletter of the Euro Working Group Multicriteria Decision Aiding 3 (1993)
9. Bouyssou, D., Vincke, P.: Relations binaires et modélisation des préférences. Technical report SMG, IS-MG 2003/02 (2003)
10. Bouyssou, D., Marchant, T., Pirlot, M., Tsoukias, A., Vincke, V.: Evaluation and decision models with multiple criteria: stepping stones for the analyst. *International Series in Operations Research and Management Science*, vol. 86. Springer, Boston (2006)
11. Brans, J.P., Vincke, P.: A preference ranking organization method. *Management Science* 31(6), 647–656 (1985)
12. Brans, J.P., Mareschal, B.: PROMETHEE V: MCDM Problems with Segmentation Constraints. *INFOR* 30(2), 85–96 (1992)
13. Brans, J.P., Mareschal, B.: PROMCALC and GAIA: A new decision support system for Multicriteria Decision Aid. *Decision Support Systems* 12, 297–310 (1994)
14. Brans, J.P.: The space of freedom of the decision maker modelling the human brain. *European Journal of Operational Research* 92(3), 593–602 (1996)
15. Brans, J.P., Macharis, C., Kunsch, P.L., Chevalier, A., Schwaninger, M.: Combining multicriteria decision aid and system dynamics for the control of socio-economic processes. An iterative real-time procedure. *European Journal of Operational Research* 109(2), 428–441 (1998)
16. Brans, J.P., Mareschal, B.: *Prométhée - GAIA: une méthode d'aide à la décision en présence de critères multiples*. Statistiques et Mathématiques Appliquées. Ellipses, Editions de l'Université Libre de Bruxelles, Paris (2002)
17. Brans, J.P., Mareschal, B.: PROMETHEE Methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 163–196. Springer, Boston (2005)
18. De Smet, Y., Vincke, P.: L'aide à la décision, *IT-Scan* 7 (2002)
19. De Smet, Y., Montano Guzman, L.: Towards multicriteria clustering: An extension of the k-means algorithm. *European Journal of Operational Research* 158(2), 390–398 (2004)
20. Dyer, J.S.: MAUT - Multiattribute Utility and Value Theories. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)
21. De Smet, Y., Nemery, P., Selvaraj, R.: An exact algorithm for the multicriteria ordered clustering problem. *Omega* 40(6), 861–896 (2006)
22. Doumpos, M., Grigoroudis, E. (eds.): *Multicriteria decision aid and artificial intelligence: Theory and applications*. John Wiley and Sons (to appear)
23. Figueira, J., Greco, S., Ehrgott, M.: *Multiple Criteria Decision Analysis: State of the Art Survey*. Springer, Boston (2005)
24. Figueira, J., Mousseau, V., Roy, B.: ELECTRE methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)
25. Hayez, Q., De Smet, Y., Bonney, J.: D-SIGHT: a new decision making software to address multi-criteria problems. *International Journal of Decision Support Systems Technologies* 4(4) (2012)
26. Keeney, R.L.: *Value-Focused Thinking*. Harvard University Press, Cambridge (1992)

27. Kilgour, D.M., Chen, Y., Hipel, K.W.: Multiple Criteria Approaches to Group Decision and Negotiation. In: Ehrgott, M., Figueira, J., Greco, S. (eds.) *Trends in Multiple Criteria Decision Analysis*. International Series in Operations Research and Management Science, vol. 142, pp. 317–338. Springer (2010)
28. Korhonen, P.: Interactive methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)
29. Neves, L.P., Dias, L.C., Antunes, C.H., Martins, A.G.: Structuring a MCDA model using SSM: A case study in Energy Efficiency. *European Journal of Operational Research* 199(3), 834–845 (2009)
30. Luce, R.D.: Semiorders and a theory of utility discrimination. *Econometrica* 24, 178–191 (1956)
31. Malczewski, J., Rinner, C.: *Multicriteria Decision Analysis in Geographic Information Science*. Springer (to appear)
32. Mareschal, B.: Weight stability intervals in multicriteria decision aid. *European Journal of Operational Research* 33(1), 54–64 (1988)
33. Mareschal, B., Brans, J.P.: Geometrical representations for MCDA: the GAIA module. *European Journal of Operational Research* 34, 69–77 (1988)
34. Markowitz, H.M.: Portfolio Selection. *The Journal of Finance* 7(1), 77–91 (1952)
35. Roy, B.: Il faut désoptimiser la Recherche Opérationnelle. *Bulletin de l'AFIRO* 7 (1968)
36. Roy, B.: Decision-aid and decision-making. *European Journal of Operational Research* 45, 324–331 (1990)
37. Roy, B.: *Multicriteria Methodology for Decision Aiding*. Kluwer Academic Publishers, Dordrecht (1996)
38. Roy, B.: A French-English Decision Aiding glossary. *Newsletter of the Euro Working Group Multicriteria Decision Aiding* 3 (2000)
39. Saaty, T.L.: The Analytic Hierarchy and Analytic Network Processes for the Measurement of Intangible Criteria and for Decision-Making. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)
40. Schärli, A.: *Décider sur plusieurs critères: Panorama de l'aide à la décision multicritère*. Presses Polytechniques Universitaires Romandes, Lausanne (1985)
41. Siskos, Y., Grigoroudis, E., Matsatsinis, N.: UTA methods. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)
42. T'kindt, V., Billaut, C.: *Multicriteria scheduling: Theory, Models and Algorithms*. Springer (2006)
43. Vansnick, J.C.: L'aide multicritère à la Décision: une Activité Profondément Ancrée Dans Son Temps. *Newsletter of the Euro Working Group Multicriteria Decision Aiding* 6 (1995)
44. Vincke, P.: *L'aide multicritère à la décision, Statistiques et Mathématiques Appliquées*, Ellipses, Editions de l'université Libre de Bruxelles, Paris (1989)
45. Vincke, P.: *Multicriteria Decision-Aid*. John Wiley and Sons, New York (1992)
46. Weistroffer, H.R., Smith, C.H., Narula, S.C.: Multiple Criteria Decision Support Software. In: Figueira, J., Greco, S., Ehrgott, M. (eds.) *Multiple Criteria Decision Analysis: State of the Art Surveys*, pp. 133–162. Springer, Boston (2005)

Knowledge Harvesting for Business Intelligence

Nesrine Ben Mustapha and Marie-Aude Aufaure

Ecole Centrale Paris, MAS Laboratory
{Nesrine.Ben-Mustapha,Marie-Aude.Aufaure}@ecp.fr

Abstract. With the growth rate of information volume, information access and knowledge management in enterprises has become challenging. This paper aims at describing the importance of semantic technologies (ontologies) and knowledge extraction techniques for knowledge management, search and capture in e-business processes. We will present the state of the art of ontology learning approaches from textual data and web environment and their integration in enterprise systems to perform personalized and incremental knowledge harvesting.

Keywords: ontology, semantics, ontology learning, knowledge engineering, business intelligence

1 Introduction

Over the past few years and with the continuous and rapid growth of information volume, information access and knowledge management, in the enterprises and on the web have become challenging. Besides, the growth rate of data repositories has been accelerated to the point that traditional Knowledge Management System (KMS) no longer provides the necessary power to organize and search knowledge in an efficient manner.

Business Intelligence(BI) solutions offer the means to transform data into information and capture knowledge through analytical tools for the purpose of enhancing decision making. Analytical tools are quite dependent on knowledge representation, capture and search. Despite the progress on these analytic tools, there are many challenges related to knowledge management that should be tackled. We argue that these issues are due to the lack of integrating the engineering of business' semantics in the foundation of BI solutions. Therefore, the improvements on knowledge engineering, search and capture offer new opportunity for building enhanced semantic BI solutions.

In fact, if semantic content of resources is described by keywords or natural language or metadata based on predefined features, it is hard to manage it because of its diversity and the need for scalability. Thus, adding a semantic layer that provides common vocabulary to represent semantic contents of resources contributes to enhance knowledge management, update, sharing and retrieval among different domains. In the emerging semantic web, Information search, interpretation and aggregation can be addressed by ontology-based semantic mark-up.

In this paper, we outline the need of *semantic technologies* for business intelligence in Section 2. After studying some motivating use case, we will detail the evolution of correlated dimensions that have been of interests by academic research groups and which include *search technologies* presented in Section 4, *Ontology learning approaches* from textual data and web environment and the machine learning techniques detailed in Section 5. In Section 6, we emphasize the importance of ontology technology and search solution capabilities for semantic revolution of the BI. Finally, we conclude with a brief synthesis.

2 Need of Semantic Technologies for Business Intelligence

We remind that the main goal of this lecture resides is explicitly transfer semantic technologies from the field of academia to industry. In this section, we will outline, the central role of knowledge in Business enterprises and the motivating scenarios of integrating semantic technologies in BI processes.

2.1 Knowledge Groups in BI Environment

In business enterprises, we can distinguish five main knowledge groups:

- knowledge workers;
- knowledge exploiters;
- knowledge learners;
- knowledge explorers;
- knowledge innovators.

Knowledge workers have an important and internal role in business enterprises. They should have great communication, learning, acting and resolving skills in order to empower the strategy of planning, sharing and collaboration of the enterprise. This group focuses mainly on internal business process and knowledge.

The main priority of knowledge exploiters resides on external knowledge learning than internal one, since they focus on competition knowledge and the development of new product. In order to achieve this purpose, this group should search daily for the external knowledge about competitors strategies, client satisfaction, etc.

The knowledge learners group aims to learn knowledge in certain areas and is not able to integrate different streams of knowledge. So, it is considered slow in learning new knowledge.

The knowledge explorers group has a central role in business enterprises, since it should maintain a good balance between internal knowledge and external knowledge group.

Knowledge innovators are qualified by "aggressive learners" as they try to combine external and external knowledge learning in order to research and disseminate findings from enterprises resources.

In the next subsection, basing on real use cases that have been studied by knowledge-Web network¹, the need of semantic technologies in business enterprise is explained.

¹ <http://www.knowledgenetworks.com/>

2.2 Motivating Use Cases: Need of Semantic Technologies

We distinguish three industrial fields for which the need of semantic technologies were discussed in a survey [80], as follows:

- Service industry;
- Media field;
- Health services.

In the following subsections, we will detail problems faced in mentioned uses cases and we will outline possible semantic architectures that can be set up.

Semantic Technologies for Service Industry. We have considered in service industry two main use cases that have been studied in [80]:

- Recruitment use case;
- B2C market place for tourism.

The recruitment service of employees on the web is an emerging and important practice in many business fields. While classic appliance ways remain available (newspaper advertisements, human resource department, etc), the internet has evolved into an open recruitment space. In Germany, 50% of recruitment are expected to be made through online job posting.

Current recruitment systems (such as Monster², experteer³ and jobpilot⁴, etc.) provide abilities to publish and search vacancies in addition to posting applicant CVs. The search process on these systems is based on predefined criteria (skills, job location, domain, etc.). So, the new challenge of these systems is to improve facilities of efficiently filling open job vacancies with the suitable candidates.

In other words, an automatic matching between job offers and job seekers information can be a good solution to overcome this challenge. In this kind of solution, an exact matching of words will not bring remarkable advantages. Integrating *semantic representation* of filled data from job seeker and job provider and *semantic search solution* can cover open issues by purposing the semantic web as technological solution.

According to this use case, we can imagine a possible architecture that integrates an ontology-based support for (Fig 11):

- expressing relationships between job characteristics and candidate qualifications (knowledge base);
- semantic querying of job offers or suitable candidates;
- learning knowledge from the web (metadata crawler);
- semantic matching.

² <http://www2.monster.fr/>

³ <http://www.experteer.fr/>

⁴ <http://www.fr.jobpilot.ch/>

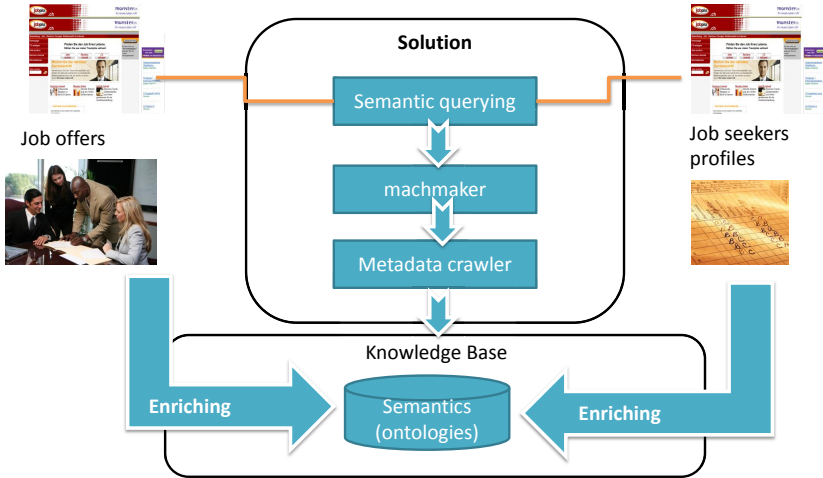


Fig. 1. Semantic solution for recruitment use case

Besides, in tourism domain which is a network business, business process relies on a number of stakeholders to develop and deliver products and services. Networks associated with Web 1.0 and Web 2.0 are confronted by serious challenges since limited interpretability is provided. In fact, dedicated sites for regional tourism have substitute the knowledge workers by content management capabilities. The choice of research criteria on B2C marketplace for tourism is limited to predefined suggestions. They are also based on pre-packaged offers.

Meantime, travel consumers are now asking for more complex packages involving many itineraries and engaging extensively through online searches in order to meet their information needs. With the current systems based on web 1.0, two main problems are inhibiting the achievement of these new challenges, which are:

- static management of web site content;
- static search process with no personalization.

Therefore, to provide a personalized service including an integrated cost of the involved services (hotels, restaurants, train, plane, geo-localization), new requirements should focus on providing:

- a web content aggregation platform;
- Dynamic exploitation of content, service providers and personalized data;
- Geo-localization;
- on-line personalized tourism packages.

Semantic Technologies for Media Field. With the continuous growing of multimedia databases, it becomes crucial to manage and maintain this huge data set. Classic solutions include faster hardware and sophisticated algorithms.

Rather a deeper understanding of media data is needed to perform the multi-media content organization, reuse and distribution.

Since the semantic web is based on machine-processable knowledge representation, there is growing interest on:

- semantic annotation of multimedia content;
- knowledge extraction from media data;
- semantic search, navigation and reasoning capabilities.

On one hand, some projects such as the aceMedia project⁵ focuses on discovering and exploiting knowledge inherent to the content in order to make searched content more relevant to the user.

On other hand, others project of news aggregation such as Neofonie⁶ are focusing on integrating semantic technologies in order to perform automatic integration and processing of news sources through a thematic clustering techniques.

Semantic Technologies for Health Services. In the context of health care, lack on data management capabilities might lead to a dramatic restructuring of the service and cost model. Doctors may ask for remote diagnosis in order to access to the accumulated knowledge of every known example of your symptoms, and your entire medical history from the time you were born.

With the digitalization of medical and health information, the doctor will be able to access to the records of all your prior treatments, including heterogenous data: images, test results, drug reactions and practitioner opinions. Therefore, he can act quickly in order to determine the suitable medications.

However, in the most occurring cases, health care organizations such as hospitals may have several information completely dispersed and not easily reused for other organizations. The main challenges are that:

- Data should be integrated independently from the data type (structured or unstructured source);
- Large health insurance companies use a cognos data warehousing solution to administrate its data;
- Business data are stored in various machines and don't share the same data formats.

Consequently, only manual search over data sources is available. For this reason, introducing common terminology for health care data and solving problems of updating data are requested. A semantic solution will involve three main actors:

- Data architect;
- Knowledge engineer;
- knowledge explorers.

The idea is to build ontologies that can be used for integrating heterogenous data marts into a single consolidate datawarehouse, as shown by figure 2.

⁵ www.aceMedia.org

⁶ <http://www.newsexpress.de>

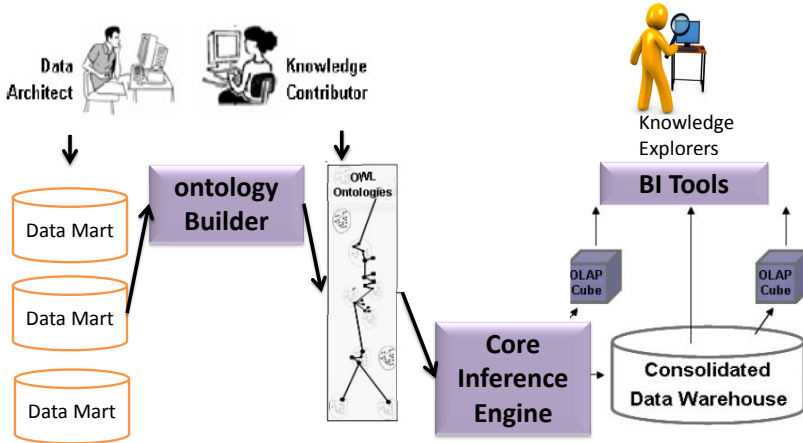


Fig. 2. Semantic solution for health use case

The overall challenge of these use cases resides in identifying where knowledge is located, how to leverage it for business purpose by harvesting knowledge from enterprise sources and from competitor events and how to manage it in an optimal way. Therefore, semantic knowledge representation is the key for the development of present intelligent systems. In the next subsection, correlated dimensions to the evolution of semantic technology are discussed since it is important to understand how these technologies have appeared in order to be able to choose the adequate technique for a given challenge faced by an industrial organization.

2.3 Correlated Dimensions Affecting Semantic Technologies

As stated in Figure 3, the main dimensions that affect the evolution of Business Intelligence Solutions are mainly:

- Semantic technologies;
- Structure of the web;
- Search methods;
- Knowledge engineering approaches.

These dimensions are quite correlated to each other and pave the way towards Business Intelligence 2.0.

Search technologies have evolved from simple keyword searching to relevance ranking (like Google). Text mining, also called text analysis, analyzes unstructured content to better determine the context and meaning of the information in relation to the search terms while relevancy ranking looks at popularity and linkages to other documents. For instance, IBM can search unstructured data sources, use text mining to extract relevant information, and load appropriate contents back into data warehouses.

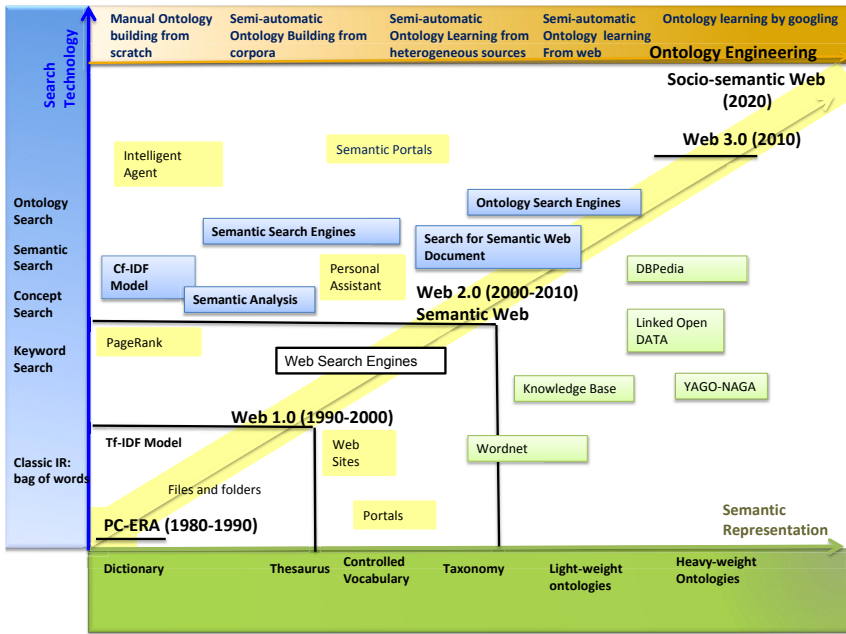


Fig. 3. Dimensions affecting Business Intelligence evolution

With the emergence of the semantic web, *Knowledge Representation* methods have evolved from dealing with controlled vocabularies, dictionaries to managing domain ontologies. Domain ontologies have become essential for managing increasing resources (contents) and promoting their efficient use for many software areas such as Bioinformatics [1], educational technology systems [2], E-learning [3], ontologies for commerce and production organization (TOVE [4] and Enterprise [5]), museums and cultural heritage and physical systems, etc.

3 Evolution of Semantics: From Dictionaries to Ontologies

As shown in Figure 3, The Knowledge Representation area has known several levels of formalization before the incoming semantic web. This is having continuously direct impact on the progress of Information Retrieval and Knowledge Engineering areas.

3.1 Levels of Knowledge Specification

Several levels of knowledge formalization can be identified, from controlled vocabularies to heavy ontologies [6], as represented in Figure 4:

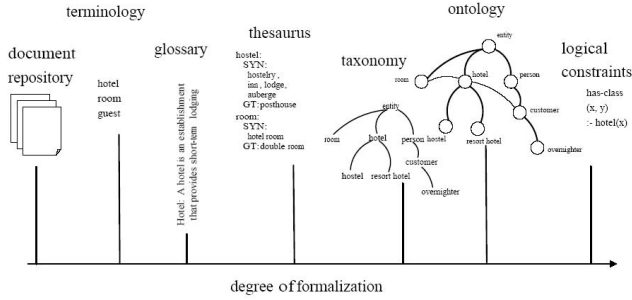


Fig. 4. Evolution of knowledge formalization

- **Controlled vocabularies:** is a set of terms defined by domain experts. The meaning of words is not necessarily defined and there is no logical organization between the terms. The vocabulary can be used in order to label documents contents. Catalogs are examples of controlled vocabularies.
- Another potential specification is a **glossary** (a list of terms and meanings): the meanings are specified by natural language statements. This provides more information since humans can read the natural language statements. Typically interpretations are not unambiguous and thus these specifications are not adequate for computer agents.
- A **thesaurus**: provides additional semantic with a limited number of relations such as synonymy (preferred term, term to use instead), related terms (a term more specific, more generic term, term related). These relationships may be interpreted unambiguously by computer agents, but no explicit relationships are specified, although with narrower and broader term specifications, a hierarchy can be deduced.
- **Informal Taxonomy:** provides explicit organizing categories from general concepts to specific ones. They have appeared on the web such as the hierarchy proposed by Yahoo for the categorization of domain topics. However, these hierarchies are not formal because the hierarchy of categories does not meet the strict notion of subsumption .
- Beyond informal “is-a” taxonomy, we move to **formal “is-a” hierarchies**. These include strict subclass relationships. In these systems, if A is a superclass of B, then if an object is a subclass of B, it is necessarily a subclass of A too.

With the emergence of the semantic web, an important trend of ontology-based application has appeared. Definitions and typology of ontological knowledge are detailed in the following subsections.

Ontology Structure. In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a

domain of knowledge or discourse. The representational primitives are typically classes (or concepts), attributes (or properties), and relationships (or relations among class members). The definitions of these representational primitives include information about their meaning and constraints on their logically consistent application.

The definition adopted by the community of knowledge engineering is the one proposed by Gruber who defines an ontology as “*an explicit specification, a formal shared conceptualization*” [8]. The *conceptualization* is the formulation of knowledge about a world in term of entities (objects, relations between these objects and the constraints of restrictions on these relations). The *specification* is the representation of the *conceptualization* in a concrete form using a knowledge representation language. In the literature, the definition and the structure of ontologies depend on the type of knowledge, the domain of knowledge and especially the usage of the ontology. In general, the structure of the ontology is composed of:

- a set of *concepts*;
- a set of *taxonomic relationships* between concepts;
- a set of *non-taxonomic relationships*;
- a set of *instances* assigned to each concept or relation;
- a set of *rules or axioms* that represent inferred knowledge of the domain.

These elements are described in the following subsections.

Concept. A Concept is defined as a class of objects related to a well-defined knowledge area such as the concept of “*human being*”, *tree*, *home*, etc. It is characterized by its meaning referred by “*Concept intension*” and by its group of entities referred by “*Concept extensions*”.

All the objects related to a concept build the set of its instances. For example, the term “*automobile*” refers both to the concept “*car*” as an object of type “*car*” and all objects of that type. A term can refer to several concepts related to several domains. An example of the term “*accommodation*” which refers to the concept of hosting web sites in the topic of “*creation of web pages*” and also the concept of *hotel accommodation* in the field of “*tourism*”. Similarly, the same concept can be referenced by multiple terms. This is the case of “*morning star*” and “*evening star*”, which both refer to Venus planet [7].

Relations. Within an ontology, we distinguish two main categories of relations: *taxonomic relations* and *non-taxonomic relations*. *Taxonomic* relations organize concepts in a tree structure and include two types of taxonomic relationships:

- relations of *hyponymy* or specialization generally known as “*is a relation*”. For example, an enzyme is a type of protein, which is a kind of macromolecule;
- partitive relations or *meronymy* that describe concepts which are part of other concepts.

On the other hand, *non-taxonomic relations* include:

- *locative relation* that describes the location of a concept. Example: “bed *is_located_in* bedroom”;
- *associative relations* that correspond to properties between concepts. Logical properties are associated with these relations such as transitivity and symmetry.

Defining the ontology only by concepts and relationships is not enough to encapsulate knowledge, since according to S. Staab and A. Maedche, the axioms and rules are a fundamental component of any ontology [9].

Axioms and Rules. Ontology *axioms* provide semantics by allowing ontology to infer additional statement. Ontological knowledge can be considered as facts, rules, or constraints. A fact is a true statement, not implicative. For example, the axiom “*the company E has 200 employees*” is a true statement. They are useful for defining the meaning of the components, setting restrictions on attribute values, specifying the arguments of a relationship and verifying the validity of specified knowledge.

In recent projects, *axioms* have been extended with *rules* in order to infer additional knowledge. For instance, the following rules “*if a company sells X products A and the price of each product is C then Sales revenue is C * X euros*” is used to calculate the revenue of daily sales.

Several languages have been developed to specify ontology *rules*. Among these languages, we cite mainly RuleML [10] and SWRL [11].

Figure 5 illustrates the formalization of the following rule specifying family relatedness using SWRL:

Rule: “hasParent (?x1,?x2) \wedge hasBrother (?x2,?x3) \longrightarrow hasUncle (?x1,?x3)”

Ontology Types. The literature defines four typologies according to the following dimensions:

- formalization degree of the ontology language (formal, informal, semi-formal);
- granularity degree of the ontology structure (light-weight ontology and heavy-weight ontology);
- level of completeness;
- type of domain knowledge.

Typology according to the type of domain knowledge was the most discussed one by several works. We distinguish the following ontology types:

- **Representational ontology:** includes primitives involved in the formalization of knowledge representation. We cite for example, the Frame ontology where primitive representation of language-based Frame are classes, instances, aspects, properties, relationships and restrictions;

```

<ruleml:imp>
  <ruleml:_rlab ruleml:href="#example1"/>
  <ruleml:_body>
    <swrlx:individualPropertyAtom swrlx:property="hasParent">
      <ruleml:var>x1</ruleml:var>
      <ruleml:var>x2</ruleml:var>
    </swrlx:individualPropertyAtom>
    <swrlx:individualPropertyAtom swrlx:property="has_brother">
      <ruleml:var>x2</ruleml:var>
      <ruleml:var>x3</ruleml:var>
    </swrlx:individualPropertyAtom>
  </ruleml:_body>
</ruleml:_head>
  <swrlx:individualPropertyAtom swrlx:property="has_uncle">
    <ruleml:var>x1</ruleml:var>
    <ruleml:var>x3</ruleml:var>
  </swrlx:individualPropertyAtom>
</ruleml:_head>
</ruleml:imp>

```

Fig. 5. Rule specification

- **Top-level ontology (also known as upper ontologies)** specifies concepts that are universal, generic, abstract and philosophical. It provides a set of general concepts from which a domain ontology can be constructed. Examples of existing upper ontologies include SUMO (Suggested Upper Merged Ontology) [\[7\]](http://suo.ieee.org/SUO/SUMO/index.html), the CYC ontology [\[8\]](http://www.cyc.com), and SUO 4D ontology [\[9\]](http://suo.ieee.org/SUO/SUO-4D/index.html);
- **Lexical ontology**: is an ontology describing linguistic knowledge, which models the meaning of words by using ontological structure. Examples of lexical ontologies are WordNet [\[12\]](#) and HowNet [\[13\]](#);
- **Domain Ontology** is tied to a specific domain which can be extended from upper ontology. Examples of domain ontologies include MENELAS in the medical field, ENGMATH for mathematics and TOVE in the field of enterprise management;
- **Ontology of tasks** [\[14\]](#): used to conceptualize specific tasks such as diagnostic tasks, planning tasks, design tasks, configuration and solving problems tasks;
- **Application ontology** defines the structure of knowledge necessary to accomplish a particular task.

⁷ <http://suo.ieee.org/SUO/SUMO/index.html>

⁸ <http://www.cyc.com>

⁹ <http://suo.ieee.org/SUO/SUO-4D/index.html>

4 New Trends of Search Paradigm

Information Retrieval (IR) research has moved from syntactic IR to semantic IR. In syntactic IR, terms are represented as sequences of characters and IR process is based on computation of string similarity. The progress made by knowledge representation and the semantic web languages areas has contributed to the development of semantic IR systems. Instead, terms are represented as concepts and IR is performed through the computation of semantic relatedness between concepts.

4.1 Semantic Web Search in Web 2.0

Several classifications of search engines for the semantic web have been proposed in the literature. Indeed, in [15], the authors distinguish:

- *Document oriented* search engines;
- *Entity oriented* search engines;
- *Multimedia search* engines;
- *Relation Oriented Search*;
- Search Engine based on *semantic analysis*.

Based on semantic search survey presented in [16], we distinguish two types of search engines for the semantic web (Figure 6):

- ontology search engines;
- semantic search engines: the use of contextual information (represented by domain ontologies and metadata) is one of the key aspects for these engines.

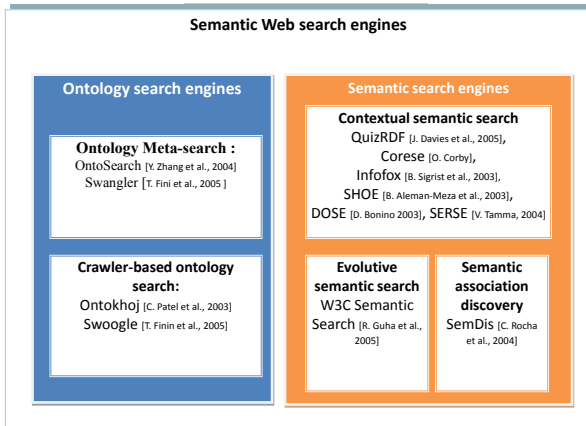


Fig. 6. Semantic web search Engine categories

The two classifications are quite complementary. Ontology search includes entity oriented search and relation oriented search. Semantic search includes the other categories.

Ontology Search Engines. We distinguish two categories of ontology search engines:

The first type corresponds to engines providing specific types of files (such as RSS, RDF, Owl) and enabling to search only by the name of files or by using some options like *filetype*. For example, in [17], *OntoSearch* engine transmits the user request to Google to search for a specific type of file and uses a visualization tool that allows the user to run her research and to show results. In [18] a technique called *Swangling* is used for this purpose. This technique offers the translation of RDF triples into strings to be transmitted to a traditional search engine. The main problem of these systems is that a lot of available semantic web documents (ontologies) are ignored. In fact, it is not obvious to collect all ontologies in the web just by using *filetype* command within existing commercial search engines.

The second type refers to crawler-based ontology search engines. The idea of these system is to build a specific crawler which is used to find Semantic Web Documents (SWD) on the web, index them and acquire some metadata about them. Ontokhoj [19] and Swoogle [18] are two crawler-based ontology search engines. By using these engines users can search for special class, property and entities.

Semantic IR Engines. The following groups can be distinguished:

- Contextual search engines;
- Evolutionary Search engines;
- Semantic association discovery engines.

Contextual Search Engines. The ultimate goal of these engines is to increase the performance of traditional search engines (especially in regard to measures of precision and recall). The use of contextual information (represented by a domain ontology and metadata) is one of the main aspects. Usually after a traditional search process, matching RDF graphs is used to obtain better results.

We distinguish seven major components: crawler, documents annotator, indexer, query formulation module, query annotation module, search module and display module. Various approaches and solutions for each of these sub-problems have been proposed [3]. It should be stressed that a very limited number of engines include all the components listed above. The quality of the results depends heavily on ontologies used. The main problem of these search engines relies with the fact that their use is limited to specific domains (represented by domain ontologies).

The best known examples are: OWLIR [20], QuizRDF [21], InWiss [22], Corese [23], SHOE [24], DOSE [25], OntoWeb [26], SERSE [27].

Evolutionary search engines. In the second group employing semantic web techniques, the objective is to accumulate information on a subject that we seek. This type of search engines is a response to a well-known problem: the automatic collection of information on a domain. The originality of these engines

is the use of external metadata (eg. CDnow, Amazon, IMDB). They are usually employed a conventional search engine and provide regular information in addition to the original results: W3C Semantic Search [28] and ABC [29].

Semantic association discovery. Search engines in the third group try to find semantic relationships between two or more terms: the aim is to find various semantic relations between the terms of entries (usually two) and then rank the results based on semantic distances. Compared to other categories, the engines dedicated for discovering semantic associations are linked to higher layers of semantic web architecture (logic and trust). SemDis [28] is an example of this group.

5 Progress in Ontology Engineering Research

The methodologies proposing manual ontology building, also known as “*from scratch*” were among the first works done in the field of ontology engineering. It consists in conceiving a process of ontology building in the absence of *a priori* knowledge (hence the meaning of the English term “*from scratch*”). Several authors have proposed many approaches based on learning techniques in order to improve the automation of this process.

The notion of learning reinforces the idea of ontology construction on the basis of *a priori* knowledge. This allows the automation of the ontology enrichment by using learning techniques. According to Maedche and Staab, there are as many ontology learning approaches as types of data sources [30]. We distinguish ontology learning approaches from texts, from dictionaries [31], from knowledge bases [32], from semi-structured schema [33] and relational data [34]. In this section, we are interested mainly in approaches related to ontology learning from web (including texts).

5.1 Ontology Learning Approaches

Ontology learning (OL) is defined as an approach of ontology building from knowledge sources using a set of machine learning techniques and knowledge acquisition methods. OL from texts is a specific case of OL from web and has been widely used in the community of knowledge engineering since texts are semantically richer than other data sources. These approaches are generally based on the use of textual corpora. This one should be a representative of the domain for which we are trying to build ontology. By applying a set of text mining techniques, a granular ontology is enriched with discovered concepts and relations from textual data. In such approach, human intervention is required to validate the relevance of learned concepts and relations.

In the last decade, with the enormous growth of online information, the web has become as an important data source for knowledge acquisition due to its huge size and heterogeneity. This led to mainly five categories of OL approaches:

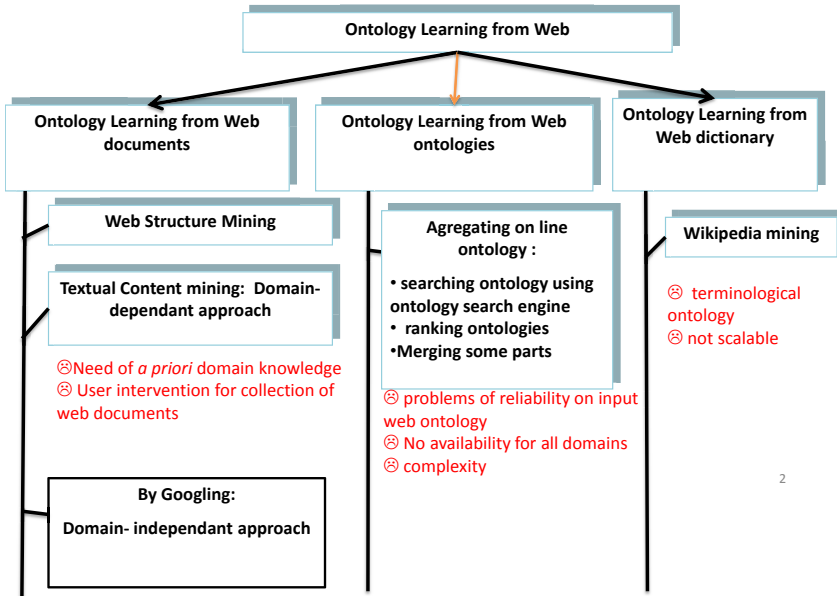


Fig. 7. Ontology Learning Approaches from web

- Ontology learning based on web content mining (texts);
- Ontology learning based on web structure mining;
- Ontology learning from web dictionary;
- Ontology learning from web ontologies;
- Ontology learning by googling.

Ontology Learning Based on Web Content Mining (Texts). OL approaches from texts have widely interested the ontology engineering community. These approaches are based on machine learning techniques applied to texts. Ontology learning process from texts consists generally in enriching a small ontology called “minimal” or “granular” ontology with new discovered concepts and relationships from input texts (corpora or web content document). This is in particular the work of: [35] [36] [39] [40] [41] [42] [43] [44] [45] [46].

By using a set of text mining techniques, knowledge contained in texts is projected to the ontology by extracting concepts and relations. We distinguish mainly five categories of techniques:

- Linguistic techniques [39] and lexico-syntactic patterns [38];
- Clustering techniques and / or classification techniques [35] [36] [67];
- Statistical techniques [47] [49] ;
- Association rule-based techniques [9];
- Hybrid ones.

Ontology Learning Based on Web Structure Mining. Furthermore, others researchers were interested to study the structure of a growing number of web pages. The underlying assumption behind **web structure mining-based** is that the noun phrases appearing in the headings of a document as well as the document's hierarchical structure [50] can be used to discover taxonomic relations between concepts.

Several systems supporting this approach analyze input documents' heading structure, extract concepts from headings and builds a taxonomical ontology. [51] defines an approach for an automated migration of data-intensive web sites into the semantic web. It is based on the extraction of light ontologies from structured resources such as XML Schema or relational database schemata and consists in building light ontologies from conceptual database schemas using a mapping process. This process provides the conceptual metadata of annotations that are automatically created from the database instances.

Besides, in [52] an approach called "Tango" uses the analysis of tables in web pages for the generation of ontologies. In these works, the main difficulty resides on the interpretation of the HTML structure that cannot reflect the semantics of documents. Human intervention is still necessary to validate the resulted ontologies.

Ontology Learning from Web Ontology. With the development of standards and tools supporting the semantic web vision, harvesting ontological files on the web has been the first step towards achieving true ontology reuse for ontology learning. The idea about online **ontology building from web ontology** has widely been explored by several works [51] [53] [54]. However, the objective was mainly to enable users to reuse or import whole ontologies or ontology modules. They provided no support for ranking available ontologies, or for extracting and merging the ontology parts of interest, or event for evaluating the resulting ontology. In [53], a framework for integrating multiple ontologies from structured documents into a common ontology is used. A universal similarity paradigm reflecting the implicit cohesion among the ontologies is presented. Ontology alignment and construction methods are applied.

Other approaches use ontology search engines or ontology meta-search engines to build ontologies by aggregating many searched domain ontologies. There is an increasing number of online libraries for searching and downloading ontologies. Examples of such libraries are Ontolingua, Protege, and DAML. Few search engines have recently appeared that allow keyword-based search for online ontologies, such as Swoogle and OntoSearch.

In [54], the proposed approach consists in searching online ontologies for certain concepts, ranking the retrieved ontologies according to some criteria, then extracting the relevant parts of the top ranked ontologies, and merging those parts to acquire the richest domain representation as possible.

We don't deny that these approaches can easily lead to obtain many domain ontologies but some problems still remain. In fact, we still worry about many issues:

- Existing web ontology are not sufficiently consistent to be used;
- the availability of ontologies to be reused in terms of number and domain variety;
- the quality of output ontology depends on the quality of input ontologies;
- the use of ontology searching, ontology ranking, ontology mapping, ontology merging, and ontology segmentation methods makes this approach more complex.

Ontology Building from Web Dictionary. “*Wikipedia mining*” is a research area recently addressed. In [55], a construction method based on Wikipedia mining is proposed. By analyzing 1.7 million concepts on Wikipedia, a very large scale ontology (called “YAGO”) which has more than 78 million associations was built. To avoid natural language processing (NLP) problems, structure mining is applied to web-based dictionaries [55].

Other Hybrid Approaches. In [30], an approach combining heterogeneous sources of information and various processing techniques associated with each type of data source was proposed in order to improve the identification of potential useful knowledge. First, it extracts the core vocabulary to the domain using a parsing process. The underlying idea of the method is that the combination of all these additional sources of evidence improves the accuracy of the OL process. Thus, the extracted terms are analyzed at five different levels: chunk, statistical, syntactical, visual and semantic level. The experimental results obtained by processing a set of HTML documents belonging to two domains, *Universities* and *Economics*, have shown the potential benefit of its use to learn or enrich ontologies following an unsupervised learning approach.

5.2 Generic Ontology Learning Process from Texts

The extraction process starts from the raw text data (text document in natural language) to obtain the final ontology knowledge representation. It includes the following steps [56]:

- Term extraction;
- Synonym extraction;
- Concept discovery;
- Taxonomic relation learning;
- Non-taxonomic relation learning.

Term Extraction. A term is a semantic unit and can be simple or complex. The terms are extracted using several techniques including statistical analysis [57], use of patterns (regular expressions), linguistic analysis (identification of nominal and prepositional groups), word disambiguation [58] and interpretation of compound phrases (as in [59] using Wordnet).

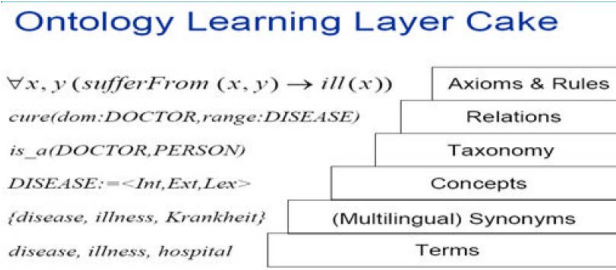


Fig. 8. Ontology Learning Process [56]

Linguistic Techniques of Term Extraction. Linguistic analysis of texts also requires the use of a grammar representing the sentence structure. We distinguish two types of grammars that mainly allow to represent the structure of a sentence in a given natural language:

- *Constituency Grammar*: this grammar is the basis of the formal theory of language used in computational linguistics. The analysis using this type of grammar is based on the position of words in the sentence and how they can be grouped.
- *Dependency Grammar*: the analysis using this grammar provide binary grammatical links between words in a sentence. When two words are connected by a dependency links relationship, we say that one is the ruler or the head and the other is a dependent. In general, the extracted relations are schematically represented by an arc between the head and the dependent.

Statistical Techniques for Term Selection. Statistical techniques are mainly based on the analysis of word co-occurrences and other parameters such as absolute frequency of a term, frequency of a term on a given field, etc. Under the assumption of Harris [60], these methods determine a score representing the relationship between two terms and retain those with scores greater than or equal to a given threshold. For example, the combination of TF* IDF measure with other methods such as *latent semantic analysis* can be used to select the domain concepts. It should be noted that these methods ignore the statistically insignificant terms.

The measures used for selection of candidate terms according to their occurrences in the corpus are as follows:

- The TF-IDF measure [61];
- The entropy [62];
- The Relevance to the domain (PD) [63];
- The Consensus in domain [63];
- The pointwise mutual information (PMI) measure [64] (formula 1).

The information contents of the concept is defined by its occurrences in the corpus as well as concepts that it subsumes. It aims to use the probability of

a concept in a corpus of documents (formula 2). The information contents of a concept c is calculated as following 47:

$$PMI(c) = -\log(p(c)) \quad (1)$$

where:

$$p(c) = \frac{freq(c)}{N} \quad \text{and} \quad freq(c) = \sum_{n \in word(c)} \quad (2)$$

Synonyms Extraction. The second step aims to identify synonyms among the extracted terms, in order to associate several words with the same concept in the same language 74. The extraction of synonyms is usually done in two ways:

- Using lexical ontologies such as Wordnet 48;
- Classification techniques which are used to group terms occurring in the same context (eg, co-occurrences of terms).

Concept Learning. Extracted terms are useful to represent the concepts of an ontology. Concept can be discovered using two techniques:

- Construction of the Topic Signature;
- Classification of concepts based on contextual properties of words.

Construction of the Topic Signature. This technique defined in 35 aims to overcome two main limits of lexical ontologies like Wordnet which are the lack of updating links between concepts and the proliferation of different meanings for each concept. This approach proceeds as follows. Firstly, information contained in existing ontology like Wordnet (synonyms of the concepts, hyponyms, antonyms, etc) is used to build requests which are used to search the relevant documents relating to one sense of a given term. The documents related to the same sense of this term are grouped together to form a collections. Secondly, the documents in each collection are processed. Words and their frequencies are extracted by using a statistical approach.

Extracted data from one collection is compared to data in other collections corresponding to the other senses of the same term. The words having a distinctive frequency for one of the collections are grouped in a list, which make up for each sense of a term, the contextual signature (*Topic signature*) generally used in the construction of summaries of texts. Thirdly, for a given word, the concepts associated with their sense are hierarchically grouped. With this intention, various signatures are compared to discover shared words and to determine intersection between the signatures. Many measures are used to calculate the semantic distance. The contextual signatures were evaluated by their application in the task of semantic disambiguation of words. These first contain considerably useful information for this task. However, the evaluation of this method by using Wordnet is not sufficient to conclude by its effectiveness in the case of domain ontology construction.

Classification of Concepts Based on Contextual Properties of Words. This technique is based on the principles of the *Distributive Semantics* which admit that “the meaning of a word is strongly correlated with the contexts in which it appears”. This assumption can be generalized to cover complex expressions instead of words. The contexts can be formalized in the shape of words vectors, as in the case of semantic signature of subject described in [65].

By using the *Topic signatures*, each concept is represented by a set of co-occurring words and their frequencies. Within this framework, several metrics of similarity, such as TF*IDF ou Chi-Square, can be used to measure the distance between various concepts. An algorithm of downward classification is described in [36] in order to extend from existing ontologies (such as Wordnet) with the new concepts. In fact, the quality of topics signatures construction described in [35] can be improved by taking in account only concepts belonging to contexts of existing ontology concepts (ie. which have syntactic relationships to the concepts in ontology). For example, it is possible to consider only the list of the verbs for which the concepts are subjects or a direct object, or to consider only the adjectives which modify the concept.

Learning Taxonomic Relations. At this step, two categories of machine learning techniques (linguistic and statistical techniques) can be used. Linguistic techniques of taxonomic relations discovery are based on the definition of lexical-syntactic patterns for extracting hyponymy relations [38]. Several statistical techniques are based on the analysis of word distribution in the corpora.

Lexico-Syntactic Patterns Related to Taxonomic Relations. Lexico-syntactic patterns are based on the study of syntactic regularities between two given concepts. Indeed, it aims to schematize the lexical and syntactic context of taxonomic relations between concepts. This mapping is a lexico-syntactic pattern and permits the retrieval of pairs of words which satisfy this relation from the corpus.

Hearst’s Patterns is the basis of several approaches. We illustrate the patterns of hyponymy relations identified by Hearst in the English language in Table 1.

Table 1. Hearst’s Patterns

| | |
|---|---|
| <i>NP such as NP, NP, ... and NP</i> | data warehousing technologies <i>such as</i> reporting, ad-hoc querying, online analytical processing (OLAP). |
| <i>Such NP as NP, NP, ... or NP</i> | <i>such</i> supervised machine learning <i>as</i> data pre-processing <i>or</i> feature selection. |
| <i>NP, NP, ... and other NP</i> | screen real estate to financial charts, indices <i>and other</i> news graphics. |
| <i>NP, especially NP, NP,... and NP</i> | Accounting, <i>especially</i> financial accounting gives mainly past information in that the events are recorded. |
| <i>NP is a NP</i> | SAS OLAP <i>is</i> a multidimensional data store engine |

In [39], the experimental evaluation of a large number of patterns was done using the Cameleon tool. The results obtained showed that the effectiveness of these patterns and their meanings depend on the corpus. Indeed, the syntactic regularities regarding the relations of hyponymy that were defined do not necessarily reflect the relevant relationships in the ontology.

Statistical Techniques for Learning Taxonomic Relations. Several statistical techniques are described in the literature for extracting taxonomic relationships between terms. They are based on analysis of co-occurrences between words in documents. The co-occurrence corresponds to the simultaneous occurrence of two words in a text (or window of n words). The set of term co-occurrences is represented by a matrix. This is then used for:

- an hierarchical grouping of words, using automatic classification methods;
- a grouping based on probability measures [66].

In this context, it is also possible to apply hierarchical clustering by using the co-occurrence matrix of words extracted from documents. In the case of a hierarchical cluster, initially, each class is composed of a term. In [66], a rule related to taxonomic relation extraction stipulates that if two concepts were referred by terms that appear in the same documents (in fact in 80 % of cases), then these two concepts are hyponyms. In other words, if a concept X subsumes a concept Y and the documents in which X appears are a subset of the documents including the word Y , then X subsumes Y . Other rules can be discovered according to the corpus. The conditional probabilities depend closely on the selected context which can be a sentence, a web page, or a web site.

Extracting Non-taxonomic Relations. Another step of ontology learning consists in discovering *non-taxonomic relations* between concepts. A non-taxonomic relation can be extracted using two main techniques:

- conceptual clustering using syntactic frames [67];
- statistical techniques.

Learning Syntactic Frames. Conceptual clustering requires a syntactic analysis of the documents from which we estimate being able to build an ontology. Classes are formed starting from the terms appearing after the same verb and the same preposition. An algorithm of conceptual clustering is applied for this purpose. One difficulty relies in labeling the relations after their discovery.

To solve this problem, two clustering algorithms: “*Asium-Best*” and “*Asium-Level*” based on the extraction of the syntactic frames were proposed by ASIUM approach in [67]. These techniques allow the discovery of non-taxonomic relations between two classes of terms. These relations are labeled according to the verb and the preposition concerned with the syntactic frame. A syntactic frame for the verb “to travel” is illustrated as following: <To travel><subject:human><by:convey>.

Initially, the parser automatically provides noun expressions associated with the verbs and the clauses. For example, starting from the following syntactic frame, classes are created:

- <To travel> (<subject: Jean>) (<in: means of transport>);
- <To travel> (<subject: David>) (<in: train>);
- <To lead> (<subject: Helene>) (<object: means of transport> ;
- <To lead> (<subject: Roland>) (<object: plane>).

The classes are successively aggregated to form new concepts hierarchies. The obtained classes are labeled by an expert to identify the concepts which they represent. The classes make up the groupings of words having the same frame: <verb> <syntactic role —preposition: name>, such as for example “ <travelling> <subject: human> <by: convey>.

The couples <syntactic role: name> or <preposition: name> are called “heads words”. Similarity measures permit to evaluate the distance between the classes, and thus to gather their dependencies based on the proportion of common “heads words” and their frequency of appearance in the documents. The method was tested on a corpus related to kitchen recipes. When the system is involved to find the couples verb-argument on 30% of the corpus, the hierarchy suggested is valid to 30%.

Statistical Techniques For Learning Non-Taxonomic Relations. The main idea of this technique is to extract noun phrases and proper names that appear frequently. These terms are called the central terms. According to the approach DOODLE II [49], co-occurring terms are proposed to be related in the ontology, and the verbs that occur in the context are proposed to be the labels of the relationship. These terms may be determined from one co-occurrence matrix in a window of n words. The advantage of this approach is that preprocessing of texts is avoided and the combination of association rules with the space of words gives better results than each technique separately used.

Finally, recent approaches which propose the use of search engines to learn ontology are described in the next section.

5.3 Ontology Capture by Googling

[68] proposes to construct an ontology by submitting the initial keywords to Google in order to retrieve web pages containing these terms. A study of several available types of search engines on the web has been carried out in order to be used in the learning process (searches web resources and calculates a score based on the number of hits) (figure 9).

The learning process proposed in this approach is based on four steps. The first one is a taxonomic learning step where the user starts to specify a keyword used as a seed for the learning process, using a web search engine. The output of this step is a one-level taxonomy and a set of verbs appearing in the same context as extracted concepts.

Secondly, non-taxonomic learning is carried out. Verb list and keywords are used as a bootstrap for the construction of domain patterns in order to submit reformulated queries to the search engine.

The third step is the recursive learning task where the two previous learning tasks are recursively executed for each discovered concept. Finally, the

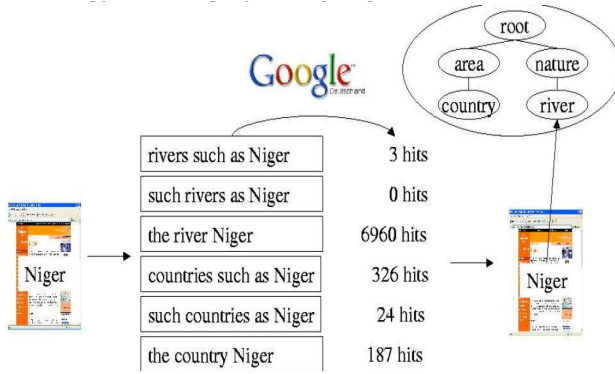


Fig. 9. Ontology leaning by googling

post-processing step consists in refining and evaluating the obtained ontology. This approach is domain independent and incremental.

These approaches led to identify three main techniques that were adapted to the web:

- statistical techniques based on term-distribution in the web;
- ontology population by Googling;
- label learning for non-taxonomic relations.

Considering the web as a massive source of knowledge, *several statistic approaches* have exploited the number of pages returned by a web search engine to estimate the probabilities of co-occurrence of terms. We consider the following notations:

- $hit(a)$ is used to denote the number of web pages containing the query returned by a search engine;
- $total_{W_{ebs}}$ denotes the total number of pages indexed by the search engine.

From a unsupervised point of view, the statistical estimation of the semantic link between concepts, as proposed in [69] [70] typically uses a measure derived from the following co-occurrence function between two terms:

$$C_k(\text{concept}, \text{candidat}) = \frac{prob(\text{concept AND candidat})_k}{prob(\text{concept}) \times prob(\text{candidat})} \tag{3}$$

The Symmetric Conditional Probability (SCP) [69] can be defined as C_2 and the Pointwise Mutual Information (PMI) [70] as $log_2 c_1$.

The probability “ $prob(a \text{ AND } b)$ ” is computed using the hit number provided by search engines, as stated by the following formula:

$$prob(a, b) = \frac{hit(a \text{ AND } b)}{total_{W_{ebs}}} \tag{4}$$

The score derived from this function was defined by Turney as follows:

$$score(\text{concept}, \text{candidat}) = \frac{prob(\text{concept AND candidat})_k}{prob(\text{concept}) \times prob(\text{candidat})} \tag{5}$$

The measures proposed by Turney were applied and evaluated in [71]. However, since the semantic content and context of words is not taken into account by these measures, limited performance is observed in [72].

Other approaches were interested in *ontology population by googling*. In fact, Gijs Geleijnse and Jan Korst [73] propose the identification of concept instances using the search engine *Google*. Queries are constructed based on lexico-syntactic patterns defined by *Hearst* [38]. A term is accepted when the number of hits (number of results returned by *Google*) exceeds a given threshold.

The same principle was also explored by [74] in order to extract taxonomic relations and attributes of concepts.

Finally, reference work on *learning non-taxonomic relation* from web has been well detailed in [74] and led to the development of the Pankow system. Pankow also relies on the idea that lexico-syntactic patterns described above can be applied not only to a text corpus, but also in the World Wide Web as in [75].

6 Ontologies for Business Intelligence

Business intelligence (BI) is defined as the process of searching, gathering, aggregating, and analyzing information for decision making.

Nowadays, Business Intelligence actors intend to bring together researchers in techniques related to conceptual modeling, **ontology engineering**, **knowledge representation**, and **Information Retrieval** for helping business developers, managers, and analysts involved in the development of BI systems to take benefits from heterogeneous data sources (unstructured and structured) and to facilitate information search.

The aim is to perform discussions on integrating ontologies, modeling languages, and search methods for the engineering of BI systems with the purpose of providing more precise information for the end-user, *bridging the gap between the dimensions that affect the evolution of Business Intelligence*.

Besides, semantic technologies advocated by semantic web [77] have been applied for BI in the context of the MUSING Project [10]. The new trend aims to develop a new generation of BI tools and modules based on *semantic-based knowledge and natural language processing (NLP) technology* to make easier gathering, merging, and analyzing information [76].

On the other hand, Ontology-based Information Extraction (OBIE) is a suitable technique for automatically extracting specific fragments from text or other sources to create records in a database or populate knowledge bases. Without an OBIE system, business analysts have to read hundreds of textual reports, web sites, and tabular data to carry out BI activities and feed BI models and tools.

In this paper we stressed the existing relation between Ontology Learning process (OL) and Ontology-based Information Extraction (OBIE) in academic research areas. This relation can be applied to the context of Business Intelligence.

¹⁰ <http://www.musing-project.eu>

In [78], authors propose a Semantic Business Intelligence (SBI) architecture that incorporates many features that distinguish it from the existing information management solutions and research. Their work aims at enabling the integration of business semantics, heterogeneous data sources, and knowledge engineering tools in order to support a smarter decision making.

Besides, the CUBIST project ¹¹ (Combining and Uniting Business Intelligence and Semantic Technologies) aims to explore standard approaches known from Formal Concept Analysis (FCA) in order to manage the complexity of the visualizations of concepts (for example, by condensing/clustering the resulting concepts, restrict visualizations by means of sub-dividing data, or filtering data in combination on other semantic query forms like faceted search) in the context of BI.

7 Conclusion

The improvement on knowledge engineering, capture and search have contributed to tackle knowledge management in the context of BI. These research areas are quite correlated and can affect positively the development of enhanced semantic Business Intelligence Tools. This paper aims to make these correlation more explicit. A state of the art about knowledge representation, recent ontology engineering approaches and semantic search engine are detailed.

On the base of analyzing ontology-based search engines presented in Section 4, we have identified the following problems:

- **the scalability of ontologies:** which makes difficult of handling several domain ontologies being used in semantic BI tools.
- **the problem of query reformulation with the use of several domain ontologies:** this is due to the usual mapping problems between query terms, ontological concepts and terms existing in documents. Indeed, identifying the ontological fragment that can be relevant for query reformulation depends strongly on the structure of the ontology. The use of the superclasses of the key concepts or the attributes in the the query reformulation task does not necessarily improve the relevance of search results. For this reason, the context of ontology-based BI applications including data type , ontology usage, users preferences is important to take into account in the ontology building process.

These problems are quire related to the progress made by ontology engineering approaches which have been widely described in this paper.

Approaches for building ontologies from online ontologies described in subsection 5 are based on the use of ontology search engines, the classification and the aggregation of the resulting ontologies. These approaches can be easily integrated in the semantic BI architecture but several problems inhibit us to continue exploring this idea, including:

¹¹ www.cubist-project.eu/

- inconsistency of the existing ontologies in online libraries;
- absence of ontologies related to several domain on the web (especially business domain)
- complexity of ontology classification, ontology alignment, merging and segmentation.

In addition, works proposing the construction of ontologies from online dictionaries allowed obtaining very large terminological ontologies (YAGO). These ontologies are quite useful but their update depends on the contents of these dictionaries, and they can be enriched only if these dictionaries undergo the updates.

For these reasons, we can consider the web as a complementary scalable source that is rich of continuously updated texts, and is covering all areas of knowledge. Using Ontology learning techniques based on googling, it will make it possible to build an integrated BI solution for incremental ontology learning. On the other side, ontology learning techniques can be applied to unstructured content, representing 80% of enterprise data, to build specific knowledge bases and enhance the search and decision processes. We were primarily interested, in this paper, in studying approaches of ontology learning from web content, since the other approaches are based on limited data sources and do not favor the evolution of the ontologies.

References

1. Camon, E., Magrane, M., Barrell, D., Lee, V., Dimmer, E., Maslen, J., Binns, D., Harte, N., Lopez, R., Apweiler, R.: The Gene Ontology Annotation (GOA) Database: sharing knowledge in Uniprot with Gene Ontology. *Nucleic Acids Res.* 32, 262–266 (2004)
2. Boyce, S., Pahl, C.: The development of subject domain ontologies for educational technology systems. *Journal of Educational Technology and Society (ETS)* IEEE 10(3), 275–288 (2007)
3. Holohan, E., Melia, M., McMullen, D., Pahl, C.: Adaptive e-Learning Content Generation Based on Semantic Web Technology. In: *Workshop on Applications of Semantic Web Technologies for e-Learning, AIED 2005, Amsterdam, The Netherlands, July 18 (2005)*
4. Fox, M.S., Barbuceanu, M.: An Organisation Ontology for Enterprise Modeling. In: Prietula, M., Carley, K., Gasser, L. (Hrsg.) *Simulating Organizations: Computational Models of Institutions and Groups*, pp. 131–152. AAAI/MIT Press, Menlo Park, CA (1998)
5. Uschold, M., Grüninger, M.: ONTOLOGIES: Principles, Methods and 16 applications. *Knowledge Engineering Review* 11(2), 93–13 (1996)
6. Navigli, R., Velardi, P.: From Glossaries to Ontologies: Extracting Semantic Structure from Textual Definitions. *Frontiers in Artificial Intelligence and Applications* 167, 71–89 (2008)
7. Furst, F., Leclère, M., Trichet, F.: Construction d'une ontology opérationnelle: un retour d'expérience. In: Hérin, D., Zighed, D.A. (eds.) *EGC. Extraction des Connaissances et Apprentissage*, vol. 1, pp. 227–232. Hermes Science Publications (2002)

8. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* (1993); Guarino, N., Poli, R. (eds.) *Special Issue on Formal Ontology in Conceptual Analysis and Knowledge Representation*
9. Maedche, A., Staab, S.: *Ontology Learning*. In: Staab, S., Studer, R. (Hrsg.) *Handbook on Ontologies*. *International Handbooks on Information Systems*, pp. 173–190. Springer (2004)
10. Harold, B.: Design Rationale of RuleML: A Markup Language for Semantic Web Rules. In: *Proceeding of SWWS 2001*, pp. 381–401 (2001)
11. Horrocks, I., Patel-Schneider, P.F., Boley, H., Tabet, S., Grosz, B., Dean, M.: SWRL: A Semantic Web rule language combining OWL and RuleML, W3C Member Submission (2004)
12. Miller, G.: Wordnet: A lexical database for English. *CACM* 38(11), 39–41 (1995)
13. Dong, Z., Dong, Q.: *HowNet and the Computation of Meaning*. World Scientific (2006)
14. Chandrasekaran, B., Josephson, J.R., Benjamins, V.R.: Ontology of tasks and methods. In: *Proceedings of the 11th Workshop on Knowledge Acquisition Modeling and Management*, Banff, Canada, pp. 20–26 (1998)
15. Wang, W., Payam, M., Barnaghi, A.B.: Search with meanings: An overview of semantic search systems. *Journal of Communications of SIWN* 3, 76–82 (2008)
16. Esmaili, K.S., Abolhassani, H.: A categorization scheme for semantic web search engines. In: *4th ACS/IEEE Int. Conf. on Computer Systems and Applications, AICCSA 2006*, pp. 171–178. IEEE (2006)
17. Zhang, Y., Vasconcelos, W., Sleeman, D.: *OntoSearch: An ontology search engine*. In: *Proceedings of the 24th SGAI International Conference on Innovative, Techniques and Applications of Artificial Intelligence*, Cambridge, UK, pp. 81–93 (2004)
18. Finin, T.W., Ding, L., Pan, R., Joshi, A., Kolari, P., Java, A., Peng, Y.: *Swoogle: Searching for knowledge on the semantic web*. In: Veloso, M.M., Kambhampati, S. (eds.), pp. 1682–1683. AAAI Press, The MIT Press (2005)
19. Patel, C., Supekar, K., Lee, Y., Park, E.K.: *OntoKhoj: a semantic web portal for ontology searching, ranking and classification*. In: *WIDM 2003*, pp. 58–61 (2003)
20. Shah, U., Finin, T., Joshi, A., Cost, R.S., Mayfield, J.: *Information Retrieval on the Semantic Web*. In: *10th International Conference on Information and Knowledge Management*, McLean, Virginia, USA, pp. 461–468 (2002)
21. Davies, J., Weeks, R., Krohn, U.: *QuizRDF: Search technology for the Semantic Web*. In: *WWW 2002 Workshop on RDF and Semantic Web Applications*, Hawaii, pp. 133–143 (2002)
22. Priebe, T.: *INWISS - Integrative Enterprise Knowledge Portal. Demonstration at the 3rd International Semantic Web Conference, ISWC 2004*, Hiroshima, Japan, pp. 33–36 (2004)
23. Corby, O., Dieng-Kuntz, R., Faron-Zucker, C.: *Querying the semantic web with corese search engine*. In: Lopez de Mantaras, R., Saitta, L. (eds.) *ECAI*, pp. 705–709. IOS Press (2004)
24. Heflin, J., Hendler, J.: *Searching the web with shoe*. In: *AAAI 2000 Workshop on AI for Web Search*, pp. 35–40 (2000)
25. Bonino, D., Corno, F., Farinetti, L.: *DOSE: A distributed open semantic elaboration platform*. In: *ICTAI*, pp. 580–588. IEEE Computer Society (2003)

26. Spyns, P., Oberle, D., Volz, R., Zheng, J., Jarrar, M., Sure, Y., Studer, R., Meersman, R.: *OntoWeb - A Semantic Web Community Portal*. In: Karagiannis, D., Reimer, U. (eds.) *PAKM 2002*. LNCS (LNAI), vol. 2569, pp. 189–200. Springer, Heidelberg (2002)
27. Tamma, V., Blacoe, I., Lithgow-Smith, B., Wooldridge, M.: *SERSE: Searching for Semantic Web Content*. In: Lopez de Mantaras, R., Saitta, L. (eds.) *Proceedings of the Sixteenth European Conference on Artificial Intelligence, ECAI 2004*, pp. 63–67 (2004)
28. Rocha, C., Schwabe, D., de Aragao, M.P.: *An hybrid approach for searching in the semantic web*. In: *Proc. of 13th Intl. World Wide Web Conf., WWW 2004*, pp. 374–383 (2004)
29. Halaschek-Wiener, C., Aleman-Meza, B., Arpinar, I.B., Sheth, A.P.: *Discovering and Ranking Semantic Associations over a Large RDF Metabase*. In: *30th International Conference on Very Large Data Bases, Toronto, Canada*, pp. 1317–1320. Morgan Kaufmann (2004)
30. Maedche, A., Staab, S.: *Ontology Learning for the Semantic Web*. *IEEE Intelligent Systems, Special Issue on the Semantic Web* 6(2), 72–79 (2001)
31. Jannink, J.: *Thesaurus Entry Extraction from an On-line Dictionary*. In: *Proceedings of Fusion 1999, Sunnyvale, CA* (1999)
32. Suryanto, H., Compton, P.: *Discovery of Ontologies from Knowledge Bases*. In: *Proceedings of the First International Conference on Knowledge Capture*, pp. 171–178. The Association for Computing Machinery, New York (2001)
33. Papatheodrou, C., Vassiliou, A., Simon, B.: *Discovery of Ontologies for Learning Resources Using Word-based Clustering*. In: *EDMEDIA 2002, Copyright by AACE, Reprinted, Denver, USA* (2002)
34. Rubin, D.L., Hewett, M., Oliver, D.E., Klein, T.E., Altman, R.B.: *Automatic data acquisition into ontologies from pharmacogenetics relational data sources using declarative object definitions and XML*. In: *Proceedings of the Pacific Symposium on Biology, Lihue, HI*, pp. 88–99 (2002)
35. Agirre, E., Ansa, O., Hovy, E., Martinez, D.: *Enriching very large ontologies using the WWW*. In: *Proceedings of ECAI Workshop on Ontology Learning, ECAI 2000* (2000)
36. Alfonseca, E., Manandhar, S.: *An unsupervised method for general named entity recognition and automated concept discovery*. In: *Proc. First International Conference on General WordNet, India* (2002)
37. Ben-Mustapha, N., Baazaoui-Zghal, H., Marie-Aude, A., Ben-Ghézala, H.: *Survey on ontology learning from web and open issues*. In: *Third International Symposium on Innovation in Information and Communication Technology, ISIICT 2009, Amman, Jordan* (2009)
38. Hearst, M.A.: *Automated Discovery of WordNet Relations*. In: *Wordnet An Electronic Lexical Database*, pp. 132–152. MIT Press, Cambridge (1998)
39. Aussenac-Gilles, N., Jacques, M.-P.: *Designing and Evaluating Patterns for Ontology Enrichment from Texts*. In: Staab, S., Svátek, V. (eds.) *EKAW 2006*. LNCS (LNAI), vol. 4248, pp. 158–165. Springer, Heidelberg (2006)
40. Bachimont, B.: *Engagement sémantique et engagement ontologique conception et réalisation d'ontologies en Ingénierie des connaissances*. In: *Ingénierie des Connaissances: Évolutions Récentes et Nouveaux Défis*, ch. 19 (2000)
41. Faatz, A., Steinmetz, R.: *Ontology enrichment with texts from the WWW*. In: *Semantic Web Mining 2nd Workshop at ECML/PKDD 2002, Helsinki, Finland* (2002)

42. Hahn, U., Mark, K.: Joint knowledge capture for grammars and ontologies. In: Gil, Y., Musen, M., Shavlik, J. (eds.) *Proceedings of the First International Conference on Knowledge Capture, K-CAP 2001*, Victoria, British Columbia, Canada, pp. 68–75. ACM Press (2001)
43. Hwang, C.H.: Incompletely and imprecisely speaking: using dynamic ontologies for representing and retrieving information. In: *Proceedings of the 6th International Workshop on Knowledge Representation meets Databases, KRDB 1999*, pp. 14–20 (1999)
44. Kietz, J.U., Maedche, A., Volz, R.: A Method for Semi-Automatic Ontology Acquisition from a Corporate Intranet. In: Aussenac-Gilles, N., Biébow, B., Szulman, S. (eds.) *EKAW 2000 Workshop on Ontologies and Texts, Juan-Les-Pins, France, Amsterdam, The Netherlands. CEUR Workshop Proceedings* (2000)
45. Moldovan, D., Girju, R.: Domain-Specific Knowledge Acquisition and Classification using WordNet. In: *Proceedings of FLAIRS 2000 Conference, Orlando*, pp. 224–228 (2000)
46. Karoui, L., Afaure, M.-A., Bennacer, N.: Contextual Concept Discovery Algorithm. In: *FLAIRS-20, The 20th International FLAIRS Conference, in Cooperation with the American Association for Artificial Intelligence, Key West, Florida, May 7-9*, pp. 460–465 (2007), special track on Context in AI tools and Applications
47. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: *IJCAI*, pp. 448–453 (1995)
48. *WordNet: An Electronic Lexical Database*. MIT Press (1989)
49. Sekiuchi, R., Aoki, C., Kurematsu, M., Yamaguchi, T.: DODDLE: A Domain Ontology Rapid Development Environment. In: Lee, H.-Y. (ed.) *PRICAI 1998*. LNCS, vol. 1531, pp. 194–204. Springer, Heidelberg (1998)
50. Karoui, L., Afaure, M.-A., Bennacer, N.: Context-based Hierarchical Clustering for the Ontology Learning. In: *IEEE Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI 2006) Jointly with the 2006 IEEE/WIC/ACM International Conference on Data Mining (ICDM 2006)*, Hong-Kong, December 18-22, pp. 420–427 (2006)
51. Stojanovic, N., Stojanovic, L., Volz, R.: A Reverse Engineering Approach for Migrating Data-intensive Web Sites to the Semantic Web. In: *17th World Computer Congress*, pp. 141–154. Kluwer Academic Publishers (2002)
52. Tijerino, Y.A., Embley, D.W., Lonsdale, D.W., Ding, Y., Nagy, G.: Towards ontology generation from tables. *World Wide Web* 8(3), 261–285 (2005)
53. Manzano-Macho, D., Gomez-Pérez, A., Borrajo, D.: Unsupervised and Domain Independent Ontology Learning: Combining Heterogeneous Sources of Evidence. In: *Proceedings of the Sixth International Language Resources and Evaluation, LREC 2008*, pp. 28–30 (2008)
54. Allani, H.: Position paper: ontology construction from online ontologies. In: *International World Wide Web Conference*, pp. 491–495 (2006)
55. Nakayama, K., Hara, T., Nishio, S.: A thesaurus construction method from large scaleweb dictionaries. In: *AINA*, pp. 932–939. IEEE Computer Society (2007)
56. Christopher, B.: In: Buitelaar, P., Cimiano, P., Magnini, B. (eds.) *Ontology Learning from Text: Methods, Evaluation and Applications* (DFKI Saarbrücken, University of Karlsruhe, and ITC-irst). *Frontiers in artificial intelligence and applications*, vol. 123, IOS Press, Amsterdam (2005); Breuker, J., et al (eds.)
57. Lin, D.: Automatic retrieval and clustering of similar words. In: *Proceedings of the 17th International Conference on Computational Linguistics, Montreal, Quebec, Canada, August 10-14*, pp. 768–774 (1998)

58. Véronis, J.: Hyperlex: lexical cartography for information retrieval. *Computer Speech & Language* 18(3), 223–252 (2004)
59. Navigli, R., Velardi, P.: Learning domain ontologies from document warehouses and dedicated web sites. *Computational Linguistics* 30(2), 151–179 (2004)
60. Harris, Z.S.: *Mathematical Structures of Language*. John Wiley and Sons, New-York (1968)
61. Robertson, S.E., Jones, K.S.: Relevance weighting of search terms. *Journal of the American Society for Information Science* 27, 129–146 (1976)
62. Brini, A.H., Boughanem, M., Dubois, D.: A Model for Information Retrieval Based on Possibilistic Networks. In: Consens, M.P., Navarro, G. (eds.) *SPIRE 2005*. LNCS, vol. 3772, pp. 271–282. Springer, Heidelberg (2005)
63. Velardi, P., Fabriani, P., Missikoff, M.: Using text processing techniques to automatically enrich a domain ontology. In: *Proceedings of the ACM Conference on Formal Ontologies and Information Systems*, pp. 270–284 (2002)
64. Lebart, L., Salem, A., Berry, L.: *Exploring Textual Data*. Kluwer Academic Publishers (1998)
65. Lin, C.-Y., Hovy, E.H.: The automated acquisition of topic signatures for text summarization. In: *COLING*, pp. 495–501. Morgan Kaufmann (2000)
66. Sanderson, M., Croft, W.B.: Deriving concept hierarchies from text. In: *Proceedings of the 22nd International ACM SIGIR Conference*, pp. 206–213 (1999)
67. Faure, D., Nedellec, C.: A corpus-based conceptual clustering method for verb frames and ontology acquisition. In: *LREC Workshop on Adapting Lexical and Corpus Resources to Sublanguages and Applications*, Granada, Spain (1998)
68. Sanchez, D.: Domain ontology learning from the web. *Knowledge Eng. Review* 24(4), 413 (2009)
69. Ferreira, J.: A local maxima method and a fair dispersion normalization for extracting multi-word units from corpora. *World Trade*, 369–381 (1999)
70. Turney, P.D.: Mining the Web for Synonyms: PML-IR versus LSA on TOEFL. In: Flach, P.A., De Raedt, L. (eds.) *ECML 2001*. LNCS (LNAI), vol. 2167, pp. 491–499. Springer, Heidelberg (2001)
71. Downey, D., Broadhead, M., Etzioni, O.: Locating complex named entities in Web text. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pp. 2733–2739 (2007)
72. Lemaire, B., Denhière, G.: Effects of High-Order Co-occurrences on Word Semantic Similarities. *Current Psychology Letters - Behaviour, Brain and Cognition* 18(1) (2006)
73. Geleijnse, G., Korst, J.H.M.: Automatic ontology population by googling. In: *Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence*, pp. 120–126 (2005)
74. Cimiano, P.: *Ontology learning and population from text - algorithms, evaluation and applications*. Springer (2006)
75. Etzioni, O., Cafarella, N., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web Scale Information Extraction in KnowItAll (Preliminary Results). In: *Proceedings of the 13th International WWW Conference*, New York, USA, pp. 100–111 (2004)
76. Kuchmann-Beauger, N., Aufaure, M.-A.: A Natural Language Interface for Data Warehouse Question Answering. In: Muñoz, R., Montoyo, A., Métais, E. (eds.) *NLDB 2011*. LNCS, vol. 6716, pp. 201–208. Springer, Heidelberg (2011)
77. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American*, 34–43 (2001)

78. Sell, D., da Silva, D., Beppler, F.D., Napoli, M., Ghisi, F.B., Pacheco, R.C., Todesco, J.L.: SBI: a semantic framework to support business intelligence. In: Proceedings of the First International Workshop on Ontology-supported Business Intelligence (OBI), p. 111. ACM, New York (2008)
79. Etzioni, O., Cafarella, M., Downey, D., Kok, S., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D.S., Yates, A.: Web Scale Information Extraction in KnowItAll (Preliminary Results). In: Proceedings of the 13th International WWW Conference, New York, USA, pp. 100–111 (2004)
80. Nixon, L., Mochol, M., Jarrar, M., Dasiopoulou, S., Papastathis, V., Kompatsiaris, Y.: Prototypical Business Use Cases. Deliverable D1.1.2 (WP1.1). The Knowledge Web Network of Excellence (NoE) IST-2004-507482, Luxemburg (January 2005)

Business Semantics as an Interface between Enterprise Information Management and the Web of Data: A Case Study in the Flemish Public Administration

Christophe Debruyne¹ and Pieter De Leenheer^{2,3}

¹ Semantics Technology and Applications Research Lab,
Vrije Universiteit Brussel, Brussels, Belgium
chrdebru@vub.ac.be

² VU University, Amsterdam, The Netherlands

³ Collibra nv/sa, Brussels, Belgium
pieterdeleenheer@acm.org

Abstract. Conceptual modeling captures descriptions of business entities in terms of their attributes and relations with other business entities. When those descriptions are needed for interoperability tasks between two or more autonomously developed information systems ranging from Web of Data with no a priori known purposes for the data to Enterprise Information Management in which organizations agree on (strict) rules to ensure proper business, those descriptions are often captured in a shared formal specification called an ontology. We present the method Business Semantics Management (BSM), a fact-oriented approach to knowledge modeling grounded in natural language. We first show how fact-oriented approaches differ from approaches in terms of, amongst others, expressiveness, complexity, and decidability and how this formalism is easier for users to render their knowledge. We then explain the different processes in BSM and how the tool suite supports those processes. Finally, we show how the ontologies can be transformed into other formalisms suitable for particular interoperability tasks. All the processes and examples will be taken from industry cases throughout the lecture.

Keywords: Conceptual Modeling, Knowledge Management, Ontology Engineering, Business Semantics Management.

1 Introduction

The increasing need for reusing and sharing information across peers in global value networks demands information systems to become Web-enabled and semantically interoperable. *Semantic interoperability* is defined as “the ability of two or more autonomously developed and maintained information systems or their (computerized) components to communicate data (using Web-based standards) and to interpret the information in the data that has been communicated in a meaningful manner” [6]. Most legacy information systems were developed in a time when these requirements were non-existing. The lack of interoperability is basically due to the different underlying formal semantics. The formal *semantics* of a (computer-based) system is the

correspondence between this system and some real world as perceived by humans and usually given by a formal mapping of the system's symbols. As the real world is not accessible inside a computer, the world needs to be represented by an agreed conceptualization if we want to store and reason about semantics. Semantics are often stored in the shape of a formal (mathematical) construct. E.g., consider a particular car that is a real-world object and its license plate being a digitized reference in a database system. The formal semantics is defined by the correspondence between the car and its unique license plate.

In order for systems to semantically interoperate, one has to have a shared understanding about this formal semantics. This is usually known as an *ontology* [14]. Ontologies constitute the key resources for realizing a Semantic Web [1]. While theoretically ontologies should be perfect renderings of a real world, in practice they evolve as successive approximations of it [15]. The problem is not so much what ontologies in computer science are, but how they come to be. The construction of ontologies is guided by appropriate ontology engineering methods. Ontology engineering is an advanced form of conceptual modeling. It requires the involvement of many parties, and they should be defined such that they are useful but also reusable. Rooted in knowledge management.

In this paper, we describe to develop and maintain ontologies for Web-based semantic interoperability. We have to address approaches from two worlds here: Web of Data and Enterprise Information Management. This article is organized as follows: in Section 2 we introduce the case for mind setting. In Section 3, we provide a background in semantic interoperability, including some of the challenges. Section 4 introduces the Business Semantic Management Method, describing the formalism, framework and brief description of the two processes: semantic reconciliation and semantic application. These two processes are then described in more detail using examples from the case in Sections 5 and 6 respectively. We then conclude this paper in Section 7.

2 The Flanders Research Information Space (FRIS) Case

For a country or region in the current knowledge economy, it is crucial to have a good overview of its science and technology base to develop an appropriate policy mix of measures to support and stimulate research and innovation. Also companies, research institutions and individual researchers can profit from the information maintained in such a portal. EWI¹ thus decided to launch the Flanders Research Information Space program (FRIS) to create a virtual research information space covering all Flemish players in the field of economy, science and innovation. The current version of this portal² contains, for instance, mash-ups of data on key entities (such as person, organization, and project; and their relationships) on a geographical map. **Fig. 1** contains a screenshot of the current FRIS portal.

¹ The Department of Economy, Science and Innovation (Economie, Wetenschap en Science in dutch) of the Flemish Government <http://www.ewi-vlaanderen.be/>

² <http://www.researchportal.be/>

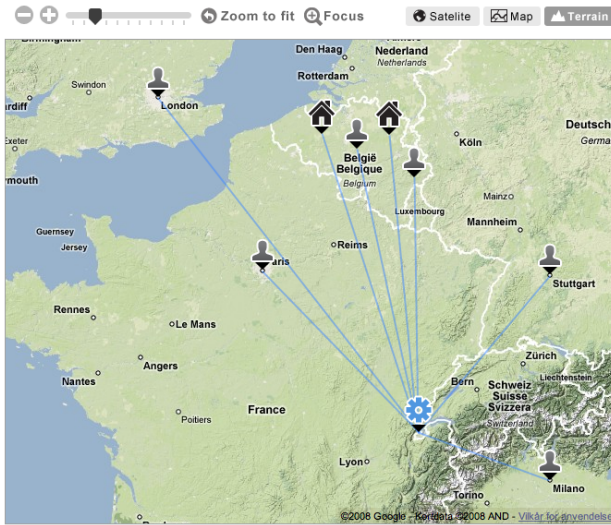
Large Hadron Collider

[Nederlands](#) [English](#)

C.E.R.N. European Organization for Nuclear Research

The Large Hadron Collider (LHC) is a gigantic scientific instrument near Geneva, where it spans the border between Switzerland and France about 100 m underground. It is a particle accelerator used by physicists to study the smallest known particles – the fundamental building blocks of all things. It will revolutionise our understanding, from the minuscule world deep within atoms to the





id: ABC_0003378

[Information](#) [Collaboration](#) [Map](#)


Content

Persons

Who has used "Large Hadron Collider"...

-  [Kouroch ABBASPOUR TEHRANI](#)
Vrije Universiteit Brussel
-  [Ahmed ABDELHAMID](#)
Vrije Universiteit Brussel
-  [AHMED ABDELKHALEK](#)
Vrije Universiteit Brussel
-  [Ademola ABOYE](#)
Vrije Universiteit Brussel

Organisations

Where "Large Hadron Collider" has been used...

-  [Elementary Particle Physics](#)
Vrije Universiteit Brussel
Tervuursevest 101, 3001 Heverlee, BE
-  [Elementary Particle Physics](#)
Vrije Universiteit Brussel
Tervuursevest 101, 3001 Heverlee, BE
-  [Elementary Particle Physics](#)
Vrije Universiteit Brussel
Tervuursevest 101, 3001 Heverlee, BE

Research projects

That has used "Large Hadron Collider"...

-  [Analysis of the top quark properties at the LHC accelerator](#)
2 Jan 2008 - 31 Dec 2011
-  [Experimental study of diffractive inter...](#)
1 Jan 2007 - 31 Dec 2011

Combining Economy, Science and Innovation for a better society
Copyright © 2008 Department of Economy, Science and Innovation, all rights reserved
Solution provided by Atria 2008, all rights reserved
[Terms of use](#) | [About your privacy](#)

Flemish Government
Department of Economy, Science and Innovation
Koning Albert II-laan 35 bus 10
B-1030 Brussel - Belgium
E-mail: iris@vlaanderen.be

Fig. 1. FRIS already provides a European map visualizing data about international cooperation between individuals, organizations and projects, e.g. in the context of the Large Hadron Collider

Another aim of FRIS is to reduce the current administrative burden for universities as they are confronted with repeatedly reporting the same information in different formats to various institutions. Universities receiving funding from the Flemish government are asked to regularly report the same information to different organizations (local and international). As there is little alignment between those reports, universities are confronted with repeatedly sending the same information in other formats, other structures or according to different classifications, not always compatible with each other. This creates a heavy administrative burden on these knowledge institutions. Universities furthermore store their information in autonomously developed information systems, adding to the complexity of the problem. As the EU also wants to track all research information in Europe, they ask all universities to report using the

Common European Research Information Format (CERIF)³, a recommendation to EU-members for the storage and exchange of current research information. If all information would be centralized and accessible in a uniform way, creating services for such reports, would greatly facilitate the reporting process.

While the CERIF model, created with Entity-Relationship (ER) [3] diagrams, allows for an almost unlimited flexibility on roles and classifications used with entities, the actual approach has shown its limitations when it comes to communicating the modeled domain knowledge to domain experts and end users. The learning curve for the domain experts to understand the ER model and translate it back to the conceptual level is quite steep [38]. For instance, the example in Fig. 2 (taken from [38]) shows the complexity of adding (multilingual) attributes to relations between core entities Person *cfPerson* and Project *cfProject*. This relation is represented by *cfPerson_Project* (linked by the two identifiers of the linked entities). In the same way, the example shows the CERIF entity *cfProject* and its relationship with the entity *cfClassification*: *cfProject_Classification*. A CERIF relationship is always semantically enriched by a time-stamped classification reference. The classification record as such is maintained in a separate entity (*cfClassification*) and allows for multilingual features (*cfClassificationTerm* and *cfClassificationDescription*). Additionally, each classification record or instance requires an assignment to a classification scheme (*cfClassificationSchemeIdentifier*). The management of the classification terms and classification schemes is organized in what is called the CERIF Semantic Layer [23].

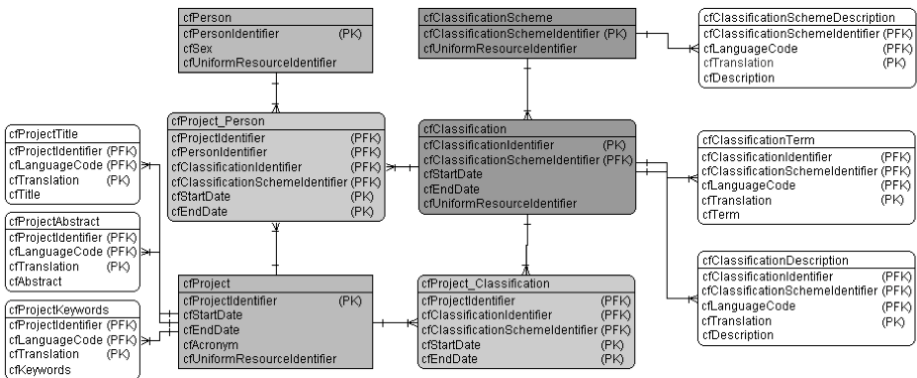


Fig. 2. The CERIF entity *cfProject* and its relationship with the entity *cfProject_Classification* (linked by the two identifiers of the linked entities). A CERIF relationship is always semantically enriched by a time-stamped classification reference. The classification record is maintained in a separate entity (*cfClassification*) and allows for multilingual features. Additionally, each classification record or instance requires an assignment to a classification scheme (*cfClassificationSchemeIdentifier*).

³ <http://cordis.europa.eu/cerif/>

Semantic mismatches occur at different levels: 1) terminology, 2) relations and 3) business rules. Due to this semantic layer, mismatches between stakeholders that need to interoperate via the CERIF standard occur at the first two levels. An example of these two mismatches on relation level is shown in Fig. 3. In this figure, two organizations use a different relation to denote that a particular researcher is the leader of a research project.

Thus, next to the conceptual complexity of the CERIF model aimed at flexibility, this flexibility also give rise to interoperability problems as heterogeneous representations for concepts and relations can be modeled.

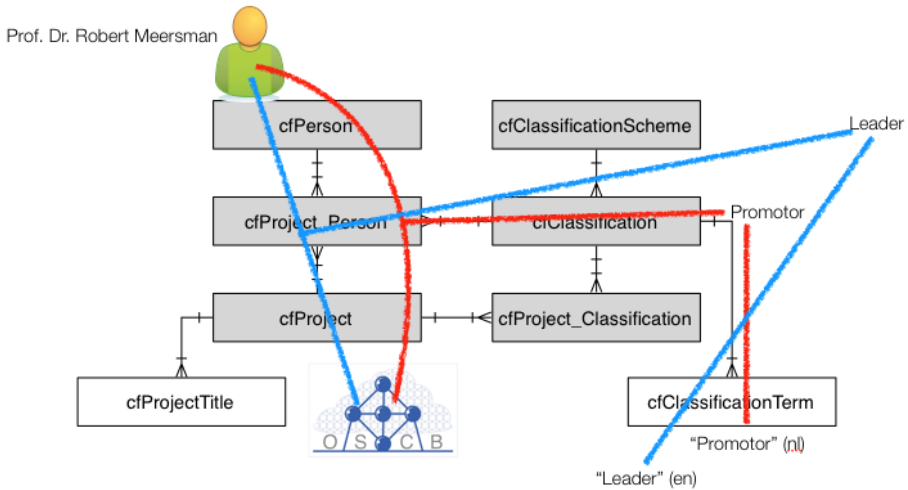


Fig. 3. Mismatch at “relation” level: two application refer to the relation between a researcher and a research project; one referring to the person as a leader of this project, the other as the promotor (“promotor” in Dutch).

To populate the FRIS portal with all information provided by the delivered CERIF files and other heterogeneous sources, needed are: 1) Consensus amongst the involved parties on a common conceptual model for CERIF and the different classifications (inside that semantic layer); 2) An easy, repeatable process for validating and integrating the data from those sources; 3) Make available the information in a generic way on the Web on which third parties can develop services as demonstrated by other Linked Data initiatives.

We furthermore have to take into account the non-technical expertise of most of the domain experts. From these requirements, it becomes clear that integrating all information and reducing the administrative burden faces some problems for which appropriate data governance methods and tools are needed. Before we present the Business Semantics Management method and its tool support, we provide the reader a background on system interoperability.

3 Background

Information systems that satisfy at least one formally specified semantic interoperability requirement are called *open information systems*. This is in contrast with *closed information systems*, where a data model represents the structure and integrity specification of the data of only the applications belonging to (often) a single enterprise. The vocabulary inside that data model in general is not a priori intended to be shared with other applications [31], i.e. the transitions caused by a closed information system are only meaningful within this system. For open information systems, however, a common vocabulary needs to be developed – and agreed upon – to which the different systems will commit to.

On the Semantic Web, a great deal of ontologies are developed in RDF(S) or OWL [39]. Both are W3C recommendations for knowledge representation languages on the Web. RDF(S) allows for the creation of simple vocabularies (concepts and relations). However, the elements provided by RDF(S) are very basic, offering little possibilities to model complex rules or constraints. The Web Ontology Language (OWL) is a family of knowledge representation languages that are more expressive than RDF(S) tailored to support some reasoning tasks such as consistency checking. Depending the “flavor” used, a particular OWL language is more expressive than another. An increase in expressiveness, however, is at the cost of efficiency or even decidability.

These ontologies are the result of knowledge management activities within a community (be it an organization, a group of organizations, etc.). Knowledge management aims at using knowledge as a production factor and comprises a range of strategies used in an organization to identify, create, represent, share, and adopt knowledge and information. Knowledge can be either elicited from individual persons or are embedded in organizations as processes or practices. Whenever two or more organizations need their autonomously developed information systems to interoperate (i.e. exchange and communicate information, do “business” together), knowledge management activities help support the group of organizations in establishing consensus on a common approximation of the real world to ensure a proper and smooth system-interoperation. Knowledge management is an important activity for both *Enterprise Information Management* (EIM) and the *Web of Data*. The first aims at satisfying the information technology needs emerging from an organization’s requirements, e.g. ensure proper business. EIM is thus a “top down” application of knowledge management. The latter aims at structuring and providing existing data in such a way (third party) services can be easily created on top of that structured information (“bottom up”).

3.1 Reusability vs. Usability of Ontologies

In many cases, ontologies contain references to the instances used in the application or application domain, and domain rules [35]. Those domain rules typically contain constraints of identity, cardinality, mandatoriness, etc. and thus restrict the semantics (i.e. interpretation) in a specific conceptualization of a particular application domain. In other words, these rules must be satisfied by any application that wishes to commit

to such an interpretation for an ontology in order for interoperability to work [14]. However, providing rules that are important for effective and meaningful interoperation between applications may (and will) limit the generality of an ontology [35]; in other words the increase of business rules decreases the generality of ontologies. This renders ontology modeling turns out to be far from trivial. Lightweight ontologies that hold none or few domain rules however are not very effective for communication between autonomously developed and maintained software systems. A requirement for different organizations in a certain domain to communicate is to have a common understanding about a relevant part of that domain. In other words, the more an ontology becomes intended for a particular application domain (more requirements, more business rules), the less general the ontology becomes.

3.2 Context of the Ontology Application

The aforementioned challenge corresponds with the variation of requirements for the Web of Data and Enterprise Information Management. The Web of Data needs meaningful annotations of data sources to enable machines to access, process and apply that information. Describing existing (legacy) data can be done with lightweight ontologies. However, as more business rules are needed to ensure proper business within the community of stakeholder, EIM will be applied to capture the requirements on how and under what conditions data will be exchanged, even up to the point how certain things have to be encoded. The Web of Data and EIM are thus residing in two different business domains and have different business drivers. The first annotates the data bottom up for third parties to develop a priori unknown services. On the other side you need top down planning with EIM to facilitate business.

The process of reaching that common understanding will involve dialogue; dialogue based on the perspectives of (ideally all) involved stakeholders. A perspective intends to capture the meaning within a given or assumed context on what the stakeholder thinks is currently relevant to the community he is in.

This semantic gap is also noticed in the discrepancy between the need for interoperability within enterprises and the actual implementation of solutions [28]. Also in the cloud computing community, the role of platform-agnostic semantic modeling is coming back (see e.g., [32]).

3.3 Requirements for a Method

Community involvement is essential for semantic interoperability. Enabling communities to develop and maintain a representation of their (business) world needs a method since reaching a common agreement between many stakeholders proves to be difficult [12]. Community involvement is crucial for facilitating the uptake and governance of, for instance, Linked Data, a set of practices for annotating and exposing data sets on the Web for which the community ultimately needs to reach an agreement on the meaning of such annotations. The Linked Data initiative relies on RDF and URI mechanisms to represent these annotations, which cannot directly map on the language of the human community. It turns out that appropriate methods for this can learn from database modeling following the principles below.

- **Technology Matures.** The non-involvement of non-tech savvy domain experts is not longer an excuse. For instance, wiki technology has been put forward as a mean to reach agreement and share knowledge about different subjects over the past decade [20]. The advantage of such technology is that anyone can add content without much technical knowledge and have already been adapted in the field of ontology engineering to enable non-technical users to create, visualize and maintain ontologies.
- **Analyzing Natural Language Discourse.** Database design methods such as NIAM [40] and ORM [18] already showed that the closer the link between human natural language communication and the system and/or business communication that results from it, the more likely such systems will work as intended by their various stakeholders. This is particularly important for interfaces where humans, systems and businesses interact, as the human discourse needs to be mapped meaningfully onto application symbols. Since people naturally communicate with words, pictures, and examples, the best way to arrive at a clear description of the domain is to use natural language, intuitive diagrams, and examples. These techniques furthermore allow scalable solutions to ontology engineering through a classical *separation of concerns* - as done in databases - by separating the schema level from the instance level. As a consequence, applications become minimally sensitive to changes in data representation.
- **Employing Legacy Data,** output reports, and interviews with domain experts as fulcrum for leveraging validation. The source (or context) of a certain fact needs to be traceable for future reference. In the case of ontology engineering: lift data models into ontologies by removing application specific context (e.g., non-conceptual identifiers such as an automatically incrementing key).

One method for collaborative ontology engineering that complies with the three principles above is Business Semantics Management.

4 Business Semantics Management

For the last twenty years, many methods have been put forward for how to develop ontologies. It seems, however, that research on methods has diminished in recent years [2]. Bergman (2010) noted that very few discrete methods exist and those that do are often older in nature [2]. He furthermore noted that most methods shared a number of logic steps from assessment to deployment, from testing to refinement.

Quite a few surveys on the state of the art on ontology engineering methods exist. Recent surveys include [34], [33] and [13]. Corcho et al. (2003) observed that there is often no correspondence between ontology building methods and tools [5]. For both the DOGMA initiative [24, 22] and Business Semantics Management (BSM), suitable tools for adequate support of these methods were developed.

BSM prescribes steps and processes for bringing a community of stakeholders together to realize the reconciliation of their heterogeneous metadata, and consequently the application of the derived business semantics patterns in partial fulfillment of well-established semantic interoperability requirements. We identify six principles of Business Semantic Management [6]:

1. **ICT Democracy.** An ontology should be defined by its owning community, and not by a single developer. In the FRIS case, the community of stakeholders contains - amongst others - the Flemish government, funding agencies, and knowledge institutions (universities).
2. **Emergence.** Semantic interoperability requirements emerge autonomously from community evolution processes. By default, business semantics serve “open” information systems, and hence the requirements and limitations for semantic interoperability cannot be entirely known before completion.
3. **Co-evolution.** Ontology evolution processes are driven by the changing semantic interoperability requirements. In contrast to waterfall-like approaches that focus on a broad design upfront, agile methods perform short milestone-driven revision iterations in order to cope with dynamic environments.
4. **Perspective Rendering.** Ontology evolution processes must reflect the various stakeholders’ perspectives. There is no generally applicable ontology, as each application will generate a contextualized model to match local needs and functionalities. Conflicts will arise from differences in how domains are perceived by the stakeholders. The different knowledge institutions in Flanders, for instance, use different classification schemes for scientific publications.
5. **Perspective Unification.** In building the common ontology, relevant parts of the various stakeholder perspectives serve as input for the unified perspective [29].
6. **Validation.** The explicit rendering of stakeholders’ perspectives allows us to capture the ontology evolution process completely, and validate the ontology against these perspectives respectively.

Ultimately, co-evolving communities with their ontology will increase overall stakeholder satisfaction.

Based on the above principles, we devised a teachable and repeatable method and system for fact-oriented BSM. The representation of business semantics is based on the DOGMA [25] ontology framework. BSM draws from DOGMA-MESS (a collaborative ontology engineering method developed on top of the DOGMA framework, first introduced in [12], further formalized in [9, 30, 6] and implemented in [4, 10]), and best practices in ontology management [19, 36] and ontology evolution [11].

4.1 Fact-Oriented

The fact-oriented paradigm that was introduced in the conceptual modeling approach NIAM (pre Object-Role Modeling). NIAM simplifies the design process by using natural language, as well as intuitive diagrams⁴, which can be populated with examples, and by examining the information in terms of simple or elementary fact types. In other words, to simplify the modeling task, stakeholders examine the information in the smallest units possible: one elementary fact at a time. By expressing the model in terms of natural concepts, like objects and roles, it provides a conceptual approach to modeling. NIAM was further refined into Object-Role Modeling, or ORM. ORM’s

⁴ In this paper, we will not go into details of ORM diagramming. More information on these diagrams can be found in [18].

rich graphic notation is capable of capturing many business rules that are typically unsupported as graphic primitives in other popular data modeling notations (e.g., role hierarchies).

Moreover, breaking down the domain into several elementary fact types reduces the problem complexity into smaller and thus more easily manageable subproblems. This leverages the potential of domain experts to effectively externalize conceptions that were not revealed otherwise [16, 17, 41].

NIAM/ORM's attribute-free approach, as opposed to frame-based techniques such as UML or (E)ER, promotes semantic stability. Semantic stability is a measure of how well models or queries expressed in the language retain their original intent in the face of changes to the application [16]. The more changes one is forced to make to a model (or query to cope with an application change), the less stable the model is. In BSM, semantic interoperability is promoted by elementary fact types that are the fundamental conceptual units of information, and are uniformly represented as relationships. How they are grouped into structures is not a conceptual issue. Given the co-evolution principle, it is critical that the underlying ontology be crafted in a way that minimizes the impact of these changes. Therefore regarding our objectives, fact-oriented models are more stable under business changes than e.g., UML or (E)ER models.

ORM models can be easily verbalized and populated for validation with domain experts, they are more stable under changes to the business domain, and they typically capture more business rules in diagram form. For instance, given the fact type:

Project, having, of, Acronym

The combination of following constraints state that a Project is totally and uniquely identified by its Acronym:

- Each Project having at most 1 Acronym
- Each Project having at least 1 Acronym
- Each Project is identified by Acronym of Project

For conceptual modeling (of information systems), the ORM method has thus several advantages over the (E)ER and UML approaches. (E)ER diagrams and UML class diagrams are closer to the final implementation, so they also have value [18] by providing "implementable" summaries of the conceptual model. In doing so, (E)ER and UML take into account constructs related to the implementation that are not relevant to the conceptualization (e.g., the difference between an attribute and a relation). The *late aggregation principle* - the act of postponing whether an object becomes an entity or an attribute until the implementation of a database is done - is well known, and fundamental, in database modeling [24] and improves the maintainability of the schema. As fact-oriented modeling techniques do not make this distinction - everything is a fact type - the modelers do not even have to consider these aspects, rendering the conceptual modeling easier.

The Semantics of Business Vocabulary and Business Rules (SBVR) [27] is an adopted standard of the Object Management Group (OMG) pushed by the business rule community and the fact-oriented modeling community. SBVR provides a fact-oriented framework for describing the semantics of terminology used in a business, business facts and business rules. The advantage of SBVR is the fact that it is an integral part of OMG's model driven architecture. SBVR uses OMG's Meta-Object Facility (MOF) [26] to provide interchange capabilities; transforming (parts) of a model into other formalisms with a MOF model (e.g., UML). MOF is essentially a set of concepts that can be used to define other modeling languages. SBVR models can be structurally linked at the level of individual facts with other MDA models based on MOF. Driven by its success in conceptual data modeling, the fact-oriented approach of SBVR provides the basis for formal and detailed natural language declarative description of complex business entities.

The structure of SBVR (illustrated in Fig. 4) allows implementing a business semantics system that takes into account the existence of multiple perspectives on how to represent concepts (by means of vocabularies), and includes the modeling of a governance model to reconcile these perspectives pragmatically (read: insofar practically necessary) in order to come to an ontology that is agreed and shared (by means of communities and speech communities) [8].

- A *semantic community* is a group of stakeholders having a body of shared meanings. Stakeholders are people representing an organization or a business unit. They already informally share knowledge via social network functionality.
- A *body of shared meanings* is a unifying and shared understanding (perception) of the business concepts in a particular domain. Concepts are identified by a URI. The scope of this body emerges from breakdowns during informal knowledge sharing.
- A *speech community* is a sub-community of a semantic community having a shared set of vocabularies to refer to the body of shared meanings. A speech community groups stakeholders and vocabularies from a particular natural language in a multi-lingual community, or from a certain technical jargon.
- A *vocabulary* is a set of terms and fact types primarily drawn from a single language to express concepts within a body of shared meanings.

The notion of vocabularies allows multi-linguality or within one language synonymous terms may refer to the same set of concepts, or a polysemous term may refer to different concept URIs depending on the vocabulary it is residing in. The following function maps a term in a vocabulary to a concept URI: $concept:Vocabulary \times Term \rightarrow URI$. For the full formalization, we refer to [9]. E.g., consider a term “student” in a Dutch vocabulary and a term “étudiant” in a French vocabulary, both meaning the same thing. Both terms are equal if and only if $concept(Dutch, student)$ and $concept(French, \acute{e}tudiant)$ refer to the same URI.

Fact-oriented models are not only suitable for modeling conceptual models for information systems. NIAM and ORM were successfully adopted for ontology engineering in a method called DOGMA.

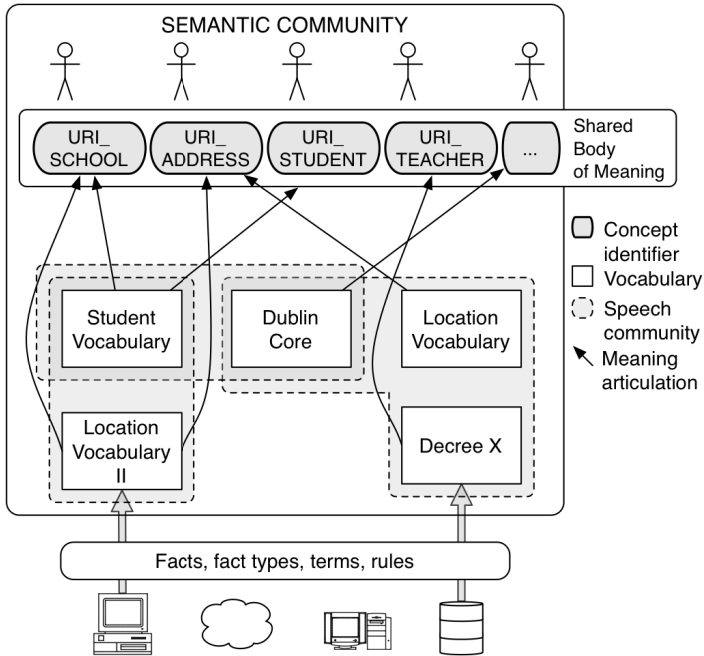


Fig. 4. The structure of business semantics: communities, stakeholders, concepts, vocabularies, facts, and rules. Speech communities are sub-communities of a semantic community having a shared set of vocabularies to refer to the body of shared meanings. Meaning articulations record those references. Applications and their symbols are then mapped onto the different fact types, terms and rules agreed upon within the community.

4.2 Development of Ontology-Grounded Methods and Applications

In the previous section we briefly described fact-oriented modeling. In this formalism, the basic knowledge building block is a fact-type; a generalization of facts encountered in the world. Initially used for developing closed information systems, it was successfully applied for modeling ontologies in the *DOGMA framework* for ontology engineering. The *DOGMA framework* that we will present in this section thus follows the fact-oriented paradigm.

Ontologies in *DOGMA* allow the application world to be associated with a lexical world relying on the fact that the knowledge building blocks expressed in natural language are easily obtained and agreed upon. These building blocks - called *lexons* [25] - only need in principle to express “plausible” fact types (as perceived by a community of stakeholders) in order to be entered into the *Lexon Base*, a repository containing large sets of such lexons. A lexon is formally described as a 5-tuple $\langle \gamma \text{ headterm, role, co-role, tailterm} \rangle$, where γ is an abstract *context identifier* pointing to a resource such as a document on the Web. The context identifier is assumed to identify unambiguously (to human users at least) the concepts denoted by the term and role labels. For example the lexon: $\langle \gamma \text{ Person, with, of, First Name} \rangle$, can be read

as: in the context γ , *Person* plays the role of *with First Name* and *First Name* plays the role of being *of Person*. The Lexon Base may contain redundant lexons, even apparently “contradictory” ones, but lexons are meant to be highly reusable and so provide semantic leverage.

The *Commitment Layer* contains ontological commitments that use a selection of lexons to annotate applications and specify constraints defining the use of the concepts in the ontology. DOGMA distinguishes two types of ontological commitments: *community commitments* and *application commitments*. The first denotes a meaningful selection of lexons, and constraints that capture the intended semantics of the data that the stakeholders want to interchange for a particular application. The latter extends the community commitment mappings describing how application symbols of one individual application commit to the ontology. The application commitment can furthermore contain additional lexons and constraints that describe how the application - as a whole - commits to the ontology [37]. Individual applications committing to the same ontology can thus have different sets of constraints. The act of selecting and constraining a meaningful selection of lexons for a particular application is called the *double articulation principle* [35]. How the lexons are used in a specific application, and the complexity associated with that use, are delegated to the ontological commitment. The use or pragmatics of lexons are thus the responsibility of the application.

Because of the resulting separation of concerns, DOGMA’s layered approach does not map one-on-one with ontologies implemented in OWL. In OWL, instances can reside next to their schema and properties are immediately constrained. DOGMA keeps the instances out of the ontology and leaves (all) interpretation and constraining of a fact type to the commitment layer. Ontologies in DOGMA are easily transformed into RDF(S) or a similar formalism and allows reasoning over domain terminology, by the late aggregation principle.

In this section, we presented the DOGMA framework to ontology engineering. What is lacking is a method for collaboratively building ontologies on top of this framework. One such method was DOGMA-MESS, in which MESS stood for Meaning Evolution Support System. We will not provide details on DOGMA-MESS, but note it was the basis for BSM. Thus in the next section, we will present the BSM method.

4.3 Business Semantics Management: Semantic Reconciliation and Application

BSM draws from best practices in ontology management [19] and ontology evolution [11]. The representation of business semantics was originally based on the DOGMA approach and provides a method and tool that enable parties to (i) obtain consensus on (the semantics of) key business terms, and (ii) evaluate this consensus uniformly in various applications throughout the organization. Respectively, BSM consists of two complementary cycles: semantic reconciliation and semantic application (see Fig. 5) where each cycle groups a number of activities.

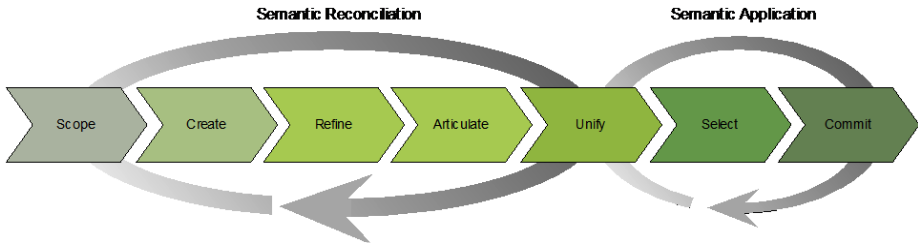


Fig. 5. Business Semantics Management consists of two complementary cycles: semantic reconciliation and semantic application. Both cycles communicate via the unify-activity.

- *Semantic Reconciliation* is the first cycle of the method. In this phase, business semantics are modeled by extracting, refining, articulating and consolidating fact types from existing sources such as natural language descriptions, existing meta-data, etc. Ultimately, this results in a number of consolidated language-neutral semantic patterns that are articulated with informal meaning descriptions (e.g., WordNet⁵ word senses). These patterns are reusable for constructing various semantic applications.
- *Semantic Application* is the second cycle. During this cycle, existing information sources and services are committed to a selection of semantic patterns. This is done by selecting the relevant patterns, constraining their interpretation and finally mapping (or committing) the selection on the existing data sources. In other words, a commitment creates a bidirectional link between the existing data sources and services and the business semantics that describe the information assets of an organization. The existing data itself is not moved nor touched.

As DOGMA's lexons and constraints are fully compatible with SBVR (supported by OMG), BSM recently adopted SBVR for representing the business domain and rules. SBVR does provide constructs that were not available in the DOGMA framework, such as support for unary fact types to represent characteristics of a business entity (e.g., Project is terminated).

The derived formal vocabularies and rules can be interpreted and used by computer systems to develop Web, software and business intelligence applications. This constitutes the semantic application of business semantics. As mentioned in a previous section, MOF provides bridges to link SBVR to OWL, RDF(S), UML, ER, etc. Via MOF, business semantics in SBVR forms the basis for forward engineering of software (i.e. UML diagrams), business intelligence (i.e. OMG common warehouse model), and Web applications (W3C RDF(S) and OWL) and vice versa: existing models can be reverse engineered to feed the BSM process.

Rather than presenting more detail on the different steps in this section, we will present the tool supporting the BSM method and work out the different steps whilst describing the tool.

⁵ <http://wordnet.princeton.edu/>

5 Semantic Reconciliation with Business Semantics Glossary

The Business Semantics Glossary (BSG) supports the semantic reconciliation processes of BSM. BSG is a Web-based software application aimed at both business as well as technical users. It lets people collaboratively manage their business semantics according to the BSM method. BSG is based on the Wiki paradigm that is a proven technique for stakeholder collaboration and is essential for evolving business semantics.

Fig. 6 illustrates the concept page (identified by a URI) in BSG for term `Project` in the BSG. The page consists of a gloss providing a natural language description; a number of fact types (e.g., `CFProject executed by CFOrganization`); a number of rules; examples; notes; and synonyms. Governance models are built-in and user roles (e.g., steward, stakeholder, as shown in Fig. 6) can be applied to distribute responsibilities and increase participation. The software takes care of the audit trails who changed what, when and why. Fine-grained permission and rights management decide which users or user groups can view/edit/monitor/ etc. different parts of the business semantics.

In this case, the BSG aims to provide a single point of reference for Flemish Public Administration's business vocabulary and rules. The different processes of semantic reconciliation are explained and exemplified with the use case in the Flemish Public Administration.

The information shown in Fig. 6 is the result of the application of the BSM method. In this section, we will describe each of the semantic reconciliation phases with examples from the FRIS case.

5.1 Scope

Scope sets out the scoped terms that are actually needed to establish semantic interoperability. Specific business drivers that want to resolve a weakness or threat in a certain application context fuel this activity. Regarding the considerations made above, a distinction between information technology or information system (IT/IS) and business contexts is made.

In an IT/IS Context, a communication breakdown may be caused by an inadequate transformation of incoming personnel data from the more than 1,500 educational institutions to the data semantics of the central salary system. The breakdown here is caused by a lack of specification of terms such as "personnel" and "salary". The derived need for manual translation (e.g., using XSLT) introduces a weakness, as defining the translation requires know-how about the respective formats. Moreover, such a translation introduces even more legacy that is difficult to interpret.

In a business context, the lack of a uniform and unambiguous meaning of the term "study area" following externally imposed rules may form a legal threat. This observation initiates another semantic reconciliation cycle where metadata related to "study area" are to be reconciled.

In any context, it is important to involve the relevant stakeholders in this process and assign them with appropriate roles and responsibilities within communities. Note that the scoping process in this paper was oversimplified; consult [7] for supporting scoping techniques.

In a previous section, we showed how SBVR foresaw structure for modeling communities (semantic and speech) and their respective communities. This was adopted in BSG and roles can be assigned to members within each community. Fig. 7 shows how these structures can be navigated in BSG.

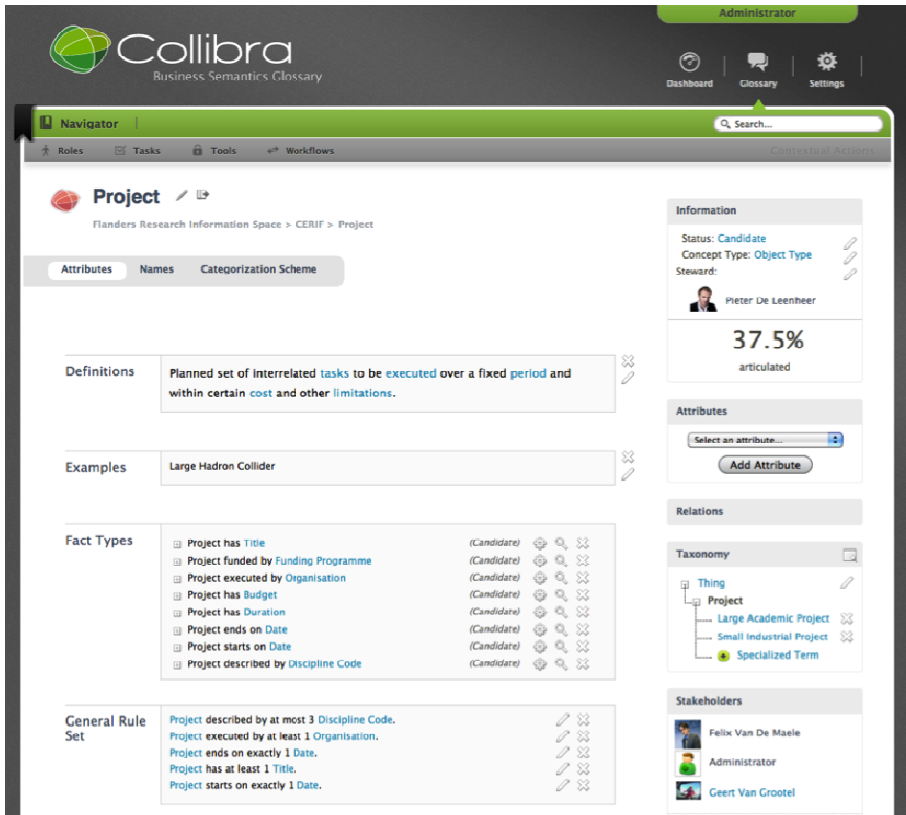


Fig. 6. Screenshot of the definition for term Project (a CERIF term) in the Project vocabulary taken from BSG that currently deployed at the Flemish public administration. Even though the concept definitions look like natural language, thanks to the underlying MOF-compliant SBVR meta-model, one can automatically generate an enterprise information model from it that provides a formal specification in UML, XSD or the like. Governance is built-in and roles can be applied to distribute responsibilities. Here, the user Pieter De Leenheer is a steward.

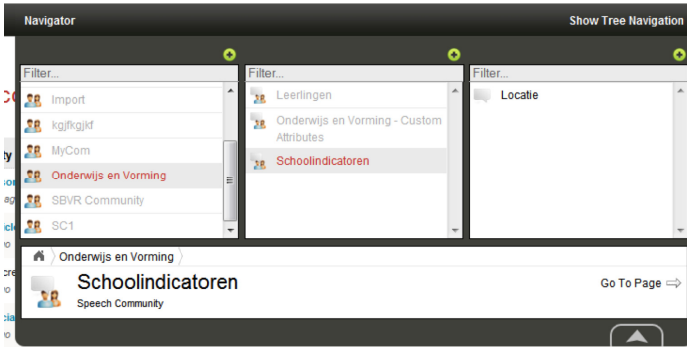


Fig. 7. Screenshot of the BSG navigator in which the user can browse through the different vocabularies. In this example the user first chooses the semantic community “Education and Training” in the first column and then the speech community “School Indicators” in the second column. This displays all the vocabularies used by that community, in this case the “Location” vocabulary can be seen.

5.2 Create

During this activity, every scoped term is syntactically defined and rules for these terms and the roles they play in their fact types are created as well. During this phase, inspiration can be drawn from existing sources (manuals, users, standards, etc.). For example, in the FRIS case, terminology may be reused from the CERIF standard:

- CFProject executed by / executes CFPerson
- CFPerson having / of Person_Name
- CFPerson having / of CFPersonAddress
- CFPersonAddress of / used in CFPersonAddress
- CFPerson affiliated with / with affiliation CFPersonAddress_UNIT
- EACH CFPerson having EXACTLY ONE Person_Name
- ...

To each scoped term, there are also certain roles appointed such as a “concept steward” and a number of relevant stakeholders. The definition is fed by implicit know-how from the involved domain experts, or by automatic extraction of facts from existing metadata (see [7] for a review of ontology extraction techniques).

5.3 Refine

During this activity, fact types (and constraints) that were created during the creation activity are refined so they are understandable to both business and technology. The refined fact-types and constraints are i) correct, ii) useful, iii) reusable, and iv) elegant. During this activity, additional fact types can be created by means of objectification (regarding a fact type as a concept, playing a role with terms of the original fact

type) or capturing missing links and relation (e.g., transforming an attribute of an entity into an attribute of a second entity related to the first entity).

In the FRIS case, the somewhat technical term `CFProj` becomes `Project` or `EmplAddr` is decomposed into a fact type `Employee is located at / locates Address`. Coding conventions can be applied here to guide the process. Below we find a set of refined fact types and constraints based on the list from the previous section:

- Project executed by / executes Organization
- Person having / of Person_Name
- Person located at / locates Address
- Person with / of Affiliation
- Organization_Unit with / of Affiliation
- EACH Person having EXACTLY ONE Person_Name
- EACH Affiliation of EXACTLY ONE Person
- EACH Affiliation of EXACTLY ONE Organization_Unit
- EACH Affiliation *a* IS IDENTIFIED BY Person with *a* AND Organization_Unit with *a*
- ...

5.4 Articulate

Create informal meaning descriptions as extra documentation. These descriptions include *definitions* and *examples* and can serve as anchoring points when stakeholders have used different terms for the same concepts (i.e., detecting synonyms). Where available, already existing descriptions can be used (e.g., the euroCRIS website on CERIF) to speed up the process and facilitate reuse.

Since multiple users may render their perspective concurrently on a term, it may be that after the refine activity some fact types and rules impose contradicting statements. During this activity, conflicts and inconsistencies are removed. Specifically designed algorithms may help here. E.g., in The Netherlands, an address is uniquely identified by a combination of postcode and house number, while in Belgium a combination of postcode, street name and street number is required. Articulating these differences is crucial in order to be able to deal with different data integrity rules during information exchanges. Fig. 8 depicts an example of a definition and example of the term “Project” in the FRIS Case.

5.5 Unification

During unification a new version of the EIM is generated, which is a “flattened” version of the BSG that is generated in a timely manner. The EIM is the product of semantic reconciliation and serves as a uniform technical specification to implement semantic applications.

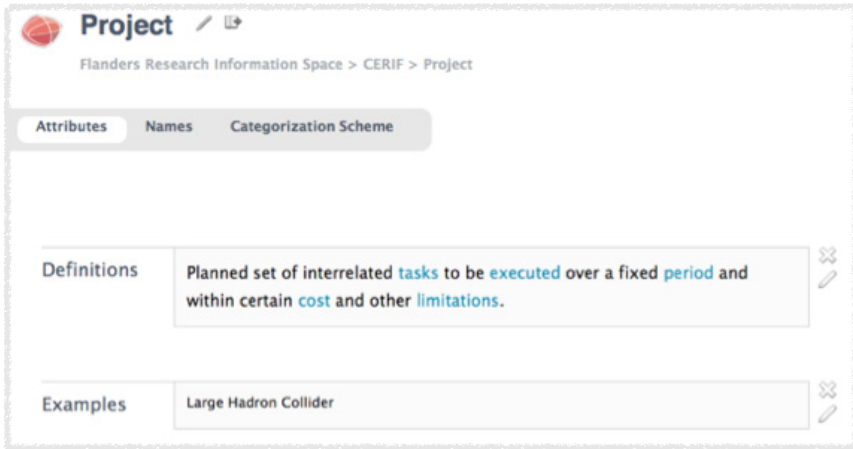


Fig. 8. Screenshot of the definition and an example for the PROJECT in the Project vocabulary of the CERIF speech community

In order to optimally consolidate equivalent groups in vocabularies, one has to check for each of these groups where redundant conceptual patterns could be combined, and note any arithmetic derivations. For instance:

- Can the same concept be a member of two concept types? If so, combine the concept types into one (unless such identities are not of interest).
- Can two objects instantiating two different concept types be meaningfully compared? Do they have the same unit or dimension? If so, combine the concept types into one.
- Is the same kind of information recorded for different entity types, and will you sometimes need to list the entities together for this information? If so, combine the entity types into one, and if necessary add another fact type to preserve the original distinction.
- Is a fact type arithmetically derivable from others?

The consolidation is finished if you were able to remove all noteworthy redundancies.

6 Semantic Application

In the previous section, we elaborated on the BSM processes that lead to descriptions of a domain, agreed upon by a community of stakeholders. These descriptions better approximate reality over time (i.e. with each iteration). Once a new version of the EIM is created, these can be applied to support the semantic interoperability requirements of that community. Through the underlying MOF framework, this EIM can be represented in many formats, such as UML, OWL, or XSD, serving a wide variety of applications.

Conceptually, we distinguish two activities: select and commit.

- **Select.** Given an application context (such as a workflow or business artifact), relevant concepts are selected from the EIM for a particular application. It may be required to add additional application-specific constraints that could not be agreed upon on the community level, or that are currently not supported by SBVR.
- **Commit.** Information systems are improved using the selected concepts. Depending on the application context, this can be implemented in different ways. Concretely, this boils down to data transformation, validation, and governance services. For example, two or more XML structures can be virtually integrated by defining XSLT transformations to a shared XSD-formatted EIM. The EIM may also be used to convert relational databases into RDF triple stores (cf. RDB2RDF initiative). Here, the application of an EIM to generate data transformation services is illustrated.

Selection and commitment thus also involves choosing the appropriate formalism for a particular task. These two activities also correspond with the creation of application commitments in the DOGMA framework for ontology engineering. The Business Semantics Studio (BSS)⁶ is a tool suite that supports these two processes. BSS provides mapping functionality to commit existing data sources and applications onto the EIM with Ω -RIDL [37]. Below are two examples of such mappings: one committing a field in a database to a concept in the EIM and another path in an XML-document. These mappings can be used to automatically generate data transformations from one format into another by generating the appropriate queries (SQL, XPath, etc.). The examples are intentionally kept simple for didactic reasons.

- A) `map "DatabaseName.TBLSchool.Street" on
Street of (/ with) Address of (/ with) School.`
- B) `map "/schools/school/street" on
Street of (/ with) Address of (/ with) School.`

The Flemish Public Administration wishes to set up a Linked Data portal for the key entities in their business-ecosystem: researchers, research projects, research organizations, etc. The Linked Data initiative aims at providing interlinked information in a representation suitable for the type of requesting agent: human readable format for users, structured data for software agents. For the latter, two simple technologies are used: URIs to identify things on the Web, and the Resource Description Framework (RDF) for describing things on the Web. To add semantics to these descriptions, ontologies materialized in RDF(S) or OWL are often used. The selection and commitment phases for this particular goal will thus include the an implementation of relevant parts of the EIM into RDF(S) or OWL. This will be described in the next section.

⁶ <http://www.collibra.com/products/business-semantics-studio>

6.1 Towards a Web of Data: Implementation in Other Formalisms

In this section, we briefly describe how (relevant parts) of the EIM is translated into other formalisms. To this end, relevant parts of the EIM need to be translated into formalisms adopted for these particular initiatives. Via MOF, parts of the EIM are also translated into – for instance – UML for the development of applications that need to be developed between stakeholders.

Even though UML is richer than SBVR for capturing some aspects of application design such as operations and components packaging, SBVR has several advantages over UML. The fact types and constraints are easily verbalized and populated (with examples) for validation with domain experts. SBVR makes no use of attributes in its base models. All fact types are represented in terms of objects playing roles. An attribute-free approach has advantages for conceptual analysis, including simplicity, stability, and ease of validation [18]. The UML specification recommends the Object Constraint Language (OCL) for formal expression of business rules, but OCL is too mathematical in nature to be used for validation by nontechnical domain experts. By design, the translation of SBVR into UML via MOF can tackle some of these issues. UML class diagrams' are valuable as the structure of those diagrams is closer to the implementation of a system. With this in mind, SBVR can be used for domain modeling and a UML diagram can be derived for the system's implementation.

Translating SBVR into OWL DL is fairly straightforward [21]. Again, not all transformation from one schema in a language into another language is *lossless*. Lossless means that both schemas are population equivalent. Fact types with arity n where $n > 2$, for instance, cannot be modeled with OWL DL. As SBVR is grounded in first order logic, it is not decidable whether a statement is provable (i.e., true under all possible interpretation). Decidability is important when one was to do reasoning, e.g., find out whether a class can have any instances or subtype inference. Many description logics are decidable fragments of first order logic, more suitable for such tasks. However, as those description logics are subsets of first order logic, translation from one to the other are not guaranteed to be equivalent.

In a first instance, EWI aimed to publish the FRIS portal data as Linked Data on the Web. In a second instance, they want to validate this data based on the business rules modeled by the community of stakeholders. To achieve the first goal, the ontology resulting from the BSM activities were translated into OWL and this OWL schema was published on the Web. The OWL schema was then used to structure, annotate and publish the information as Linked Data on the Web. This process actually corresponds with the semantic application of BSM; facts are selected to annotate the existing data source to achieve interoperability.

Fig. 9 shows a part of the generated OWL from the concept depicted in the previous figure. In this figure, we see that `Organizational_Unit` is a `Class` and instances of that class can be characterized by keywords (a `Literal`). Furthermore, an `Organizational_Unit` is composed of instances of `Person` (again a `Class`) and through the `Organizational_Unit_composed_of_Person` property. The inverse role is also specified.

```

<owl:DatatypeProperty
rdf:about="#Organizational_Unit_characterised_by_Keyword">
  <rdfs:label>characterised by Keyword</rdfs:label>
  <rdfs:domain rdf:resource="#Organizational_Unit"/>
  <rdfs:range
    rdf:resource="http://www.w3.org/2000/01/rdf-
schema#Literal"/>
<owl:ObjectProperty
  rdf:about="#Organizational_Unit_composed_of_Person">
  <rdfs:label>composed of Person</rdfs:label>
  <rdfs:domain rdf:resource="#Organizational_Unit"/>
  <rdfs:range rdf:resource="#Person"/>
  <owl:inverseOf
    rdf:resource="#Person_member_of_Organizational_Unit"/>
</owl:ObjectProperty>

```

Fig. 9. Screenshot of the OWL around Project generated by BSG. In this picture, we see that Person is a Class and Persons have roles in an organization.

The contents of the databases to be annotated can be published with off-the-shelf solutions such as D2R Server⁷. D2R Server generates an RDF description containing a mapping for transforming the content of a database into RDF triples. This mapping – also described in RDF – contains a “skeleton” RDF(S) of classes and properties that are based on the database schema. Fig. 10 below depicts a part of the generated mapping file around the table containing information around projects.

```

@prefix map: <file:///.../OSCB/d2r-server-0.7/map.n3#>.
@prefix vocab: <http://192.168.0.136:5432/vocab/resource/>.
@prefix d2rq: <http://www.wiwiss.fu-
berlin.de/suhl/bizer/D2RQ/0.1#>.
...
map:CFPROJ a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "CFPROJ/@@CFPROJ.CFPROJID|urlencode@@";
  d2rq:class vocab:CFPROJ;
  d2rq:classDefinitionLabel "EWI.CFPROJ";
...

```

Fig. 10. Part of the generated mapping file by D2R server, it maps the table CFProj to the generated CFPROJ RDF(S) class. It uses the primary key to generate a unique ID and the class definition label is taken from the table’s name.

Even though classes and properties are generated and populated with instances, these RDF triples are not semantic as they stem from one particular information

⁷ <http://www4.wiwiss.fu-berlin.de/bizer/d2r-server/>

system (its database schema). The RDF(S) skeleton is thus complemented with the generated RDF(S)/OWL classes and properties generated from the BSM ontology. The commitments described in the previous section are used as a guideline to create this alignment. Fig. 11 below shows the changes (highlighted) made on the generated mapping file with the ontology. The ontology can then be used to access the data.

```

@prefix map: <file:///.../OSCB/d2r-server-0.7/map.n3#>.
@prefix vocab: <http://192.168.0.136:5432/vocab/resource/>.
@prefix d2rq:
http://www.wiwiss.fu-berlin.de/suhl/bizer/D2RQ/0.1#>.
@prefix ont: <file:///.../Project.rdf#> .
...
map:CFPROJ a d2rq:ClassMap;
  d2rq:dataStorage map:database;
  d2rq:uriPattern "CFPROJ/@CFPROJ.CFPROJID|urlencode@@";
  d2rq:class ont:Project;
  d2rq:classDefinitionLabel "Project";
...

```

Fig. 11. Modified mapping file with the ontology exported from BSG. An extra namespace (for the exported ontology) is added and the generated classes and properties are appropriately annotated with that ontology.

To achieve the second goal, the resulting OWL file can be used for one of its popular decision problems: *classification*. Classification or instance checking corresponds with the question: “is a particular instance a member of a given concept?” Whenever we have an instance of one of EWI’s key entities (e.g., Project), it can be compared against the business rules around that concept by asking a reasoner whether this particular instance fits this class.

6.2 Full-Cycle BSM: Validation and Feedback

Once semantic applications are running, it must be possible to monitor and feed unexpected side effects or failures back, calling for a new iteration of BSM. We call this full-cycle BSM: the scope of the next version of the EIM is fed by the validation of the previous version in IT/IS contexts as well as business contexts. The BSG is the vehicle that serves the reconciliation of the newly scoped concepts.

The BSM cycle is repeated until an acceptable balance of differences and agreements is reached between the stakeholders that meets the requirements of the semantic community. Gradually, closed divergent metadata sources are replaced with metadata sources that follow an open standard, and are kept coherent via BSG.

7 Conclusions

In this paper, we presented the Business Semantics Management (BSM) method for knowledge modeling and ontology engineering. BSM was implemented in the

Flemish Public Administration for the building in the context of the Flanders Research Information Space (FRIS) program. The examples throughout this paper originate from this case.

Even though different formalisms exist for capturing certain parts of the domain, BSM's fact oriented nature, expressed in natural language enables stakeholders to quickly participate in the knowledge modeling processes. Depending on the actual goal of the community, translations or "implementations" of the fact-oriented ontology into other formalisms can be generated. We have shown how the BSM ontology was translated into OWL to publish the FRIS portal data as Linked Data on the Web.

From a high-level perspective, three different kinds of data exchange exist within large organizations: 1) Exchange of knowledge between people; 2) Exchange of understanding between people and information systems; and 3) And exchange of data between disparate information systems.

In this paper and given the requirements of the FRIS case, we focused on the third aspect. All large enterprises, however, face a semantic gap that makes all three of these exchanges extremely inefficient. The BSM method and supporting tools help in capturing the necessary semantics for rendering these exchange processes more efficient by providing a reference point for data governance questions such as: (1) what does my data mean? (2) where and how is my data utilized? (3) who is responsible for my data?

Acknowledgements. The authors would like to thank the Flemish government for the case study. This work was partially funded by the Institute for Promotion of Research and Innovation in the Brussels Capital Region and the European FP7 ACSI project.

References

1. Berners-Lee, T., Hendler, J., Lassila, O.: The Semantic Web. *Scientific American* 284(5), 35–43 (2001)
2. Bergman, M.: A brief survey of ontology development methodologies (2010), <http://www.mkbergman.com/906/a-brief-survey-of-ontology-development-methodologies/>
3. Chen, P.: The entity-relationship model: Toward a unified view of data. In: Kerr, D. (ed.) *VLDB*, p. 173. ACM (1975)
4. Christiaens, S., De Leenheer, P., de Moor, A., Meersman, R.: Business use case: Ontologising competencies in an interorganisational setting. In: Hepp, M., De Leenheer, P., de Moor, A., Sure, Y. (eds.) *Ontology Management for the Semantic Web, Semantic Web Services, and Business Applications, from Semantic Web and Beyond: Computing for Human Experience*. Springer (2008)
5. Corcho, O., Fernandez-Lopez, M., Gomez-Perez, A.: Methodologies, tools and languages for building ontologies, where is their meeting point? *Data and Knowledge Engineering* 46(11), 41–64 (2003)
6. De Leenheer, P.: *On Community-based Ontology Evolution: Foundations for Business Semantics Management*. Phd thesis, Vrije Universiteit Brussel (2009)
7. De Leenheer, P.: Ontology elicitation. In: Liu, L., Özsu, M.T. (eds.) *Encyclopedia of Database Systems*, pp. 1966–1972. Springer, US (2009)

8. De Leenheer, P., Christiaens, S., Meersman, R.: Business semantics management: A case study for competency-centric HRM. *Computers in Industry* 61(8), 760–775 (2010)
9. De Leenheer, P., de Moor, A., Meersman, R.: Context Dependency Management in Ontology Engineering: A Formal Approach. In: Spaccapietra, S., Atzeni, P., Fages, F., Hacid, M.-S., Kifer, M., Mylopoulos, J., Pernici, B., Shvaiko, P., Trujillo, J., Zaihrayeu, I. (eds.) *Journal on Data Semantics VIII*. LNCS, vol. 4380, pp. 26–56. Springer, Heidelberg (2007)
10. De Leenheer, P., Debruyne, C.: DOGMA-MESS: A Tool for Fact-Oriented Collaborative Ontology Evolution. In: Meersman, R., Tari, Z., Herrero, P. (eds.) *OTM 2008 Workshops*. LNCS, vol. 5333, pp. 797–806. Springer, Heidelberg (2008)
11. De Leenheer, P., Mens, T.: Ontology evolution: State of the art and future directions. In: [19], pp. 131–176
12. de Moor, A., De Leenheer, P., Meersman, R.: DOGMA-MESS: A Meaning Evolution Support System for Interorganizational Ontology Engineering. In: Schärfe, H., Hitzler, P., Øhrstrøm, P. (eds.) *ICCS 2006*. LNCS (LNAI), vol. 4068, pp. 189–202. Springer, Heidelberg (2006)
13. Gomez-Perez, A., Fernandez-Lopez, M., Corcho, O.: *Ontological Engineering with examples from the areas of Knowledge Management, e-Commerce and the Semantic Web*. Springer (2003)
14. Gruber, T.: Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies* 43, 907–928 (1993)
15. Guarino, N.: Formal ontology, conceptual analysis and knowledge representation. *International Journal of Human and Computer Studies* 43(5-6), 625–640 (1995)
16. Halpin, T.: Metaschemas for ER, ORM and UML data models: A comparison. *J. Database Manag.* 13(2), 20–30 (2002)
17. Halpin, T.: Comparing metamodels for ER, ORM and UML data models. In: Siau, K. (ed.) *Advanced Topics in Database Research*, vol. 3, pp. 23–44. Idea Group (2004)
18. Halpin, T.: *Information Modeling and Relational Databases*. Morgan Kaufmann, San Francisco (2008)
19. Hepp, M., De Leenheer, P., de Moor, A., Sure, Y. (eds.): *Ontology Management, Semantic Web, Semantic Web Services, and Business Applications. Semantic Web and Beyond Computing for Human Experience*, vol. 7. Springer (2008)
20. Hepp, M., Siorpaes, K., Bachlechner, D.: Harvesting wiki consensus: Using wikipedia entries as vocabulary for knowledge management. *IEEE Internet Computing* 11(5), 54–65 (2007)
21. Hodrob, R., Jarrar, M.: ORM To OWL 2 DL Mapping. In: *Proceedings of the International Conference on Intelligent Semantic Web – Applications and Services*, pp. 131–137. ACM (2010)
22. Jarrar, M., Meersman, R.: Ontology Engineering - The DOGMA Approach. In: Dillon, T.S., Chang, E., Meersman, R., Sycara, K. (eds.) *Advances in Web Semantics I*. LNCS, vol. 4891, pp. 7–34. Springer, Heidelberg (2008)
23. Jörg, B., Krast, O., Jeffery, K., van Grootel, G.: *CERIF2008XML - 1.0 Data Exchange Format Specification*, euroCRIS (2009b)
24. Meersman, R.: Towards models for practical reasoning about conceptual database design. In: *Proc. of the 2nd IFIP 2.6 Working Conference on Database Semantics, “Data and Knowledge” (DS-2)* (1986)
25. Meersman, R.A.: Semantic Ontology Tools in IS Design. In: Raś, Z.W., Skowron, A. (eds.) *ISMIS 1999*. LNCS, vol. 1609, pp. 30–45. Springer, Heidelberg (1999)
26. OMG: Meta object facility, v2.0 (2009), <http://omg.org/spec/MOF/2.0/>

27. OMG: Semantics of business vocabulary and business rules, v1.0 (2009), <http://omg.org/spec/SBVR/1.0/>
28. OVUM, INMARK: Value it (support action grant agreement no.: 216710): D3.2 final demand driven mapping report (2010)
29. Petrie, C.: Pragmatic semantic unification. *IEEE Internet Computing* 9(5) (2005)
30. Schürr, A., Nagl, M., Zündorf, A. (eds.): AGTIVE 2007. LNCS, vol. 5088. Springer, Heidelberg (2008)
31. Sheth, A., Kashyap, V.: So far (schematically) yet so near (semantically). In: Hsiao, D., Neuhold, E., Sacks-Davis, R. (eds.) DS-5. *IFIP Transactions*, vol. A-25, pp. 283–312. North-Holland (1992)
32. Sheth, A., Ranabahu, A.: Semantic modeling for cloud computing, part 2. *IEEE Internet Computing* (4), 81–84 (2010)
33. Simperl, E.P.B., Tempich, C.: Ontology Engineering: A Reality Check. In: Meersman, R., Tari, Z. (eds.) OTM 2006, Part I. LNCS, vol. 4275, pp. 836–854. Springer, Heidelberg (2006)
34. Siorpaes, K., Simperl, E.: Human intelligence in the process of semantic content creation. *World Wide Web* 13(1-2), 33–59 (2010)
35. Spyns, P., Meersman, R., Jarrar, M.: Data modelling versus ontology engineering. *SIGMOD Record Special Issue* 31(4), 12–17 (2002)
36. Spyns, P., Tang, Y., Meersman, R.: An ontology engineering methodology for DOGMA. *Applied Ontology* 3(1-2), 13–39 (2008)
37. Trog, D., Tang, Y., Meersman, R.: Towards Ontological Commitments with Ω -RIDL Markup Language. In: Paschke, A., Biletskiy, Y. (eds.) *RuleML 2007*. LNCS, vol. 4824, pp. 92–106. Springer, Heidelberg (2007)
38. Van Grootel, G., Spyns, P., Christiaens, S., Jörg, B.: Business Semantics Management Supports Government Innovation Information Portal. In: Meersman, R., Herrero, P., Dillon, T. (eds.) OTM 2009 Workshops. LNCS, vol. 5872, pp. 757–766. Springer, Heidelberg (2009)
39. Wang, T.D., Parsia, B., Hendler, J.: A Survey of the Web Ontology Landscape. In: Cruz, I., Decker, S., Allemang, D., Preist, C., Schwabe, D., Mika, P., Uschold, M., Aroyo, L.M. (eds.) *ISWC 2006*. LNCS, vol. 4273, pp. 682–694. Springer, Heidelberg (2006)
40. Wintraecken, J.: *The NIAM Information Analysis Method, Theory and Practice*. Kluwer Academic Publishers (1990)
41. Zhao, G.: AKEM: an ontology engineering methodology in ff poirot. FF POIROT Project Deliverable 6.8 (2005)

Author Index

| | | | |
|-----------------------|-----|-----------------------|-----|
| Aufaure, Marie-Aude | 177 | Le Borgne, Yann-Aël | 62 |
| Ben Mustapha, Nesrine | 177 | Lidouh, Karim | 150 |
| Ben Taieb, Souhaib | 62 | Markl, Volker | 125 |
| Bontempi, Gianluca | 62 | Pedersen, Torben Bach | 1 |
| Debruyne, Christophe | 208 | Saecker, Michael | 125 |
| De Leenheer, Pieter | 208 | Skhiri, Sabri | 103 |
| De Smet, Yves | 150 | Spies, Marcus | 78 |
| Jouili, Salim | 103 | Vaisman, Alejandro | 29 |