

UNIVERSITÀ DELLA CALABRIA

**Dipartimento di Elettronica,
Informatica e Sistemistica**

**Dottorato di Ricerca in
Ingegneria dei Sistemi e Informatica
XXIII ciclo**

Tesi di Dottorato

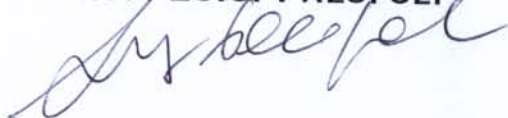
Getting Knowledge from Presentation-Oriented Documents

ERMELINDA ORO



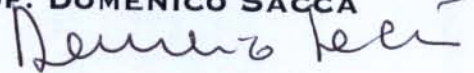
COORDINATORE

PROF. LUIGI PALOPOLI



SUPERVISORI

PROF. DOMENICO SACCÀ



ING. MASSIMO RUFFOLO



DEIS- DIPARTIMENTO DI ELETTRONICA, INFORMATICA E SISTEMISTICA
Novembre

Settore Scientifico Disciplinare: ING-INF/05

Acknowledgement

A PhD course is both a challenging and enjoyable activity. At the end I would like to thank persons that have supported me and made very special this experience.

I want to thank Andrea for his love. I know what different idea we have about work, and that you did not agree with all the time spent for research. Now we started to share more time and decisions and I love this time spent together. I hope that the research world, that is very uncertain and unstable in Italy, will allow me to show you how satisfactory it can be.

I'm forever indebted to my family. Thanks for your simplicity and strength that allow you to overcome, or at least to accept, the very difficulties of the life. I hope that you can be able to understand how much love you give each other without call it love explicitly.

Thank to my supervisor, Domenico Saccá, because he believe in me and made possible to start this PhD. I appreciate all his contributions of time and ideas to make my PhD experience productive and stimulating.

Thank to the co-relator Massimo Ruffolo, the person that, most of any other one, has taught me the basis of this work, has shared every idea with me and have made possible to arrive here.

I want to thank Steffen Staab, who gave me the opportunity to work with him and spend some months of my PhD in the University of Koblenz-Landau. You have contributed immensely to my personal and professional growth. The joy and enthusiasm he has for his research was contagious and motivational for me. I am also thankful for the excellent example he has provided as a successful charismatic person and professor. Thank to Silke, she made me feel at home, and to all the guys of the WeST group for the time spent together.

I like to remember here the new Altilia s.r.l. workgroup: Gianni, Umberto and Francesco. I hope that the group will grow because of your enthusiasm, and genuineness. So that the group will be very powerful and compact.

I'm also grateful to Antonio, Mario, Sabrina and the persons of icar and deis that supported me and made enjoyable time spent at university.

VI Acknowledgement

I like to remember here Barbara, because she are learning me the dance that creates positive energy increasing my self esteem. Thanks to all dance classmate. You all are very nice, friendly and funny.

Thank to all you because you made less hard, and indeed enjoyable this experience.

Rende,

Ermelinda Oro

November 2010

To my family, for their Love

Preface

The Web was originally conceived as a mean for sharing documents. It was thought as a set of nodes in a network storing interlinked hypertext documents that can be rendered by browsers and presented to human users. Today, the Web is the largest repository of human knowledge and acts as a worldwide collaborative platform. Everyone can produce, share and improve a very diverse spectrum of (multimedia) documents having different internal encodings. Throughout the last decade, Web search engines, such as Google and Yahoo, retrieve potentially interesting list of Web documents for answering user information needs. However, in many cases users have specific needs that cannot be answered by an ordered set of documents. Users need objects (having a clear semantics) retrieved from different documents and arranged in a comprehensible way. However, contents are distributed on the Web by two of the most diffused document formats such as the Hypertext Markup Language (HTML) and Adobe portable document format (PDF). HTML and PDF are semi and unstructured document formats conceived for presenting documents on screen or for printing respectively. In this dissertation we subsume documents encoded by HTML or PDF formats as *presentation-oriented documents* (PODs). In such kind of document the spatial arrangement of contents items produces visual cues that help human readers to make sense of document contents. So, the problem of extracting meaningful information from presentation-oriented documents can be better addressed by approaches capable to exploit the spatial structure of such kind of documents. This thesis faces such challenging problem by defining approaches that allow for querying and wrapping PODs available on the Web. In particular, in the thesis are described:

- A data models that allows for representing presentation-oriented documents independently from the document format. Such model represents both the internal hierarchical structure of PODs (e.g. the DOM), and the spatial arrangement of layouted basic data (e.g text, images, video).

- Techniques and algorithms intended to automatically recognize and extract repetitive records, tables and other typical spatial and structural pattern by using internal structures as well as spatial relations, and semantic and presentation features of PODs.
- Query languages that allow for navigating the internal structure of PODs by exploiting presentation information, layout patterns and spatial relations between data. Languages enable, also, to describe objects to extract on the base of semantic and linguistic features.

Approaches defined in the thesis are able to exploiting prior knowledge about specific search domains represented in suitable ontology languages, which enable semantic-aware information extraction.

Described novel approaches and systems allow to obtain relevant information from PODs and constitute a first steps for achieving a new generation of Web technologies capable of querying and organizing information on the Web and produce structured representations of Web contents, a point which is central in researches on the future Internet.

Rende,

Ermelinda Oro

November 2010

Contents

Part I Background and Uptake

1	Introduction and Overview	3
1.1	Problem Description and Motivations	3
1.2	Main Contributions	7
1.3	Outlook	8
1.3.1	Thesis' Structure	8
1.3.2	Reader's Guide	9
1.3.3	Publications	9
2	Background	13
2.1	Presentation-Oriented Documents – PODs	13
2.1.1	Internal Representations	14
2.1.2	Displaying PODs	20
2.1.3	Challenges	22
2.2	Web Query Languages	24
2.2.1	XPath	24
2.2.2	XQuery	28
2.3	Formal Languages	30
2.3.1	Grammars	31
2.3.2	Parsing Techniques	34
2.4	Knowledge Representation	38
2.4.1	Ontologies	38

Part II Information Extraction: State Of Art

3	Flat Text Information Extraction	49
3.1	Pattern Matching Based IE	50
3.2	Machine Learning Based IE	50

4	Web Information Extraction	53
4.1	Manual IE	53
4.2	Semiautomatic IE	53
4.2.1	Wrapper Induction Systems	54
4.2.2	GUI-based Web Data Extraction	55
4.3	Automatic IE	57
4.3.1	Visual-based Approaches	58
5	PDF Information Extraction	61
5.1	Document Understanding	61
5.2	PDF Table Recognition	62
5.3	PDF Wrapping	63
6	Semantic Information Extraction	65
6.1	Ontology-based Information Extraction System	65
7	Query Languages	67
7.1	Logic-based Query Languages	67
7.2	Spatial Query Languages	68
7.3	Multimedia Query Languages	68

Part III Information Extraction: New Approaches

8	Towards a Spatial and Semantic Information Extraction System	73
8.1	Architecture and Functionalities	73
9	Document Model for PODs	77
9.1	Qualitative Spatial Relations	78
9.2	2-Dimensional Flat Representation	84
9.3	Spatial Document Model – SDOM	86
9.4	Discussion	92
10	Automatic Information Extraction	93
10.1	PDF-TREX: Understanding PDF Documents	94
10.1.1	Recognizing and Extracting Tables	97
10.1.2	Experiments	104
10.2	SILA: Extracting Data Records from Deep Web Sites	116
10.2.1	The SILA Algorithm	118
10.2.2	Experiments	122
10.3	Discussion	129

11 Querying PODs	131
11.1 SXPath: Spatial Querying of PODs	132
11.1.1 Syntax and Semantics	139
11.1.2 Complexity Issues	144
11.1.3 Implementation and Experiments	151
11.2 Spatial Grammars	161
11.2.1 Syntax and Semantics	161
11.2.2 Complexity Issues and Experiments	167
11.3 Discussion	169
12 Semantic Information Extraction	171
12.1 Self-Describing Ontologies – SDO	172
12.1.1 Logic 2-D Representation	173
12.1.2 Object and Class Descriptors	177
12.2 XONTO-L: Logic-Based System for Extracting Objects	179
12.3 XONTO-G: Grammar-Based System for Extracting Objects ..	185
12.4 Discussion	192
13 Visual Information Extraction	193
13.1 Visual Fetuares in the SILA and SXPath System	193
13.2 Visual Features in the PDF-TREX System	196
14 Real World Applications	199
14.1 Medical Process Management	199
14.1.1 Process Modelling	199
14.1.2 Ontology-based Clinical KR Framework	202
14.2 Records Management	210
14.3 Document Understanding	211
14.4 News Paper Review	212
<hr/>	
Part IV Conclusion and Future Work	
<hr/>	
15 Conclusion and Future Work	217
15.1 Conclusion and Future Work	217
15.1.1 Content Summary	217
15.1.2 Contributions	218
References	221

List of Figures

1.1	Structure of the thesis and chapter dependencies	10
2.1	(a) An unranked tree; (b) its representation using the binary relations “firstchild” and “nextsibling”.	15
2.2	A Page of the http://www.lastfm.it/ Web Site	18
2.3	A DOM Portion of the Web Page in Fig. 2.2	18
2.4	A PDF Document about Tabular Data on Electricity Production	19
2.5	Rectangles that Bound Visualized DOM Nodes	21
2.6	Rectangles that Bound Content Items in PDF Document	22
8.1	Architecture of an Information Extraction System	74
9.1	An example of MBR	78
9.2	Pictorial Representation of the RA relations	80
9.3	Cardinal tiles	81
9.4	(a) r_1 N:NW:NE r_2 . (b) r_1 B:W:NW:N:NE:E r_2	83
9.5	RCC5 model	83
9.6	A page of Ebay Web Site with highlighted some CIs	85
9.7	A SDOM Fragment of the Web Page Portion in Fig. 11.1	88
9.8	Ordering directions	90
9.9	The SDOM building process	91
9.10	Class Diagram of Document Model Implementation	91
10.1	The Input PDF Document	95
10.2	Initial CIs obtained from the input in Fig. 10.1	96
10.3	Clusters obtained from the input in Fig. 10.1	96
10.4	Cartesian plane and final MBRs obtained from the input in Fig. 10.1	96
10.5	Logical architecture of Types of PDF DOM	97
10.6	The Running Example Page	100
10.7	Basic Content Items	101

XVI List of Figures

10.8	Elements, Segments, Lines, Text and Table Areas	102
10.9	Blocks, Rows, Columns and the Final Cells Grid	105
10.10	Full Split Error: (a) Input table; (b) output table.	108
10.11	Split Spanning Error: (a) Input table; (b) output table.	108
10.12	Full Split Error: (a) Input table; (b) output table.	109
10.13	Whole Rows and Multi-columns Merging Error: (a) Input table; (b) output table.	109
10.14	Splitting and Merging Errors occur in a same table: (a) Input table; (b) output table.	110
10.15	Additional lines and columns: (a) Input table; (b) output table.	110
10.16	Wrongly data added to boundary cells (a) Input table; (b) output table.	111
10.17	Rows and Columns Not Recognized: (a) Input table; (b) output table.	111
10.18	Input Table	112
10.19	The table 10.18 split in five tables	112
10.20	An input table having a header not closely to data cells	113
10.21	Header of the table 10.20	113
10.22	Data cells of the table 10.20	113
10.23	Merged Table Error: (a) Input tables; (b) output table.	114
10.24	(a) Input PDF document; (b) “nonexistent” recognized table.	115
10.25	Flat Structure	123
10.26	Nested Structure	124
10.27	Standard Structure	125
10.28	Html fragment with a vertical region and a horizontal region	126
10.29	Extraction of wrong records: (a) input Web page; (b) extracted records	127
10.30	Merging of records: (a) input Web page; (b) extracted records	128
10.31	Correctly extracted records and items: (a) input Web page; (b) extracted records and items	129
10.32	Extracted records with items not aligned: (a) input Web page; (b) extracted records and items	130
11.1	Rectangles that Bound Visualized DOM Nodes	133
11.2	A DOM Portion of the PDF Page in Fig. 2.4	135
11.3	A SDOM Fragment of the PDF Page in Figure. 2.4	136
11.4	Fragments of Web Pages representing friend lists of social networks (a) Bebo and (b) Care (c) Netlog, with associated DOM structures.	138
11.5	(a) Data items representing books. (b) Visualization areas. (c) DOM tree encoding of data items	138
11.6	Linear-time Data Efficiency of XPath Query Evaluation	153
11.7	Quadratic-time Complexity of SDOM Construction	153
11.8	Linear-time Query Efficiency of XPath Query Evaluation	153

11.9 Basic and inferred CIs of a data record of the Ebay Web Page (Fig. 9.6).	162
11.10The New York Times Web page.	163
11.11Results of Experiments	169
12.1 Architecture of the XONTO System	180
12.2 A sample of some transition network aimed to recognize a weather table.	182
12.3 Fragment of the OO-ATN aimed at recognizing and acquiring information about weather tables.	183
12.4 Yahoo Chicago Weather Page	186
12.5 Cartesian Plane and MBRs for the Table contained in Figure 12.4	186
12.6 Architecture of the SPO System Prototype.	187
12.7 Module 2-Dimentional Matcher of the SPO System Prototype.	189
12.8 Labeled Dependency Graph	190
12.9 A sketch of the parse tree resulting from the query <code>X:weatherForecastTable()</code> ?	190
13.1 Visual Interface of the SXPath System	194
13.2 Visual Interface of the SILA System	194
13.3 Visual Selection of a Data Record for Wrapper Learning.	195
13.4 Data Records Identified by the Visual Wrapper	195
13.5 Teh User Interface of the PDF-TREX System	197
13.6 Tables Automatically Recognized In a PDF Document	198
13.7 A Table Recognized by a Visual Selection	198
14.1 (a) The process meta-model. (b) The nodes hierarchy. (c) A clinical process for caring the breast neoplasm	201
14.2 A Visual Wrapper for an Invoice	211
14.3 The Result of the Visual Wrapping Process for an Invoice	212
14.4 The Result of the Visual Wrapping Process for an Official Bulletin	213
14.5 The Result of the Visual Wrapping Process for the NYT Web Site	213
14.6 The Result of the Visual Wrapping Process for the ANSA Web Site	214

List of Tables

2.1	Traditional XPath Axes definition	26
9.1	IA Relations.	79
9.2	RCRs as cartesian products of RA Relations	82
9.3	Topological relations mapped into RA Relations	83
10.1	Precision and Recall	105
10.2	Comparison of precision and recall results for both table areas and cells	106
10.3	Classifications for table areas	114
10.4	Classifications for cells	115
10.5	Comparison of precision and recall results for both table areas and cells	124
10.6	Classifications for Records and Items	127
11.1	Abbreviated syntax	142
11.2	Semantics of Functions in SXPath	144
11.3	Comparison between complexity bound of SXPath and XPath 1.0 for a XML document D and a query Q	151
11.4	Evaluation of the Effort Needed for Learn and Apply SXPath ..	156
11.5	Usability Evaluation of SXPath on Deep Web Pages.	157
11.6	Generality of SXPath Queries on Social Network Sites	160

Part I

Background and Uptake

*Simple things should be simple,
complex things should be possible.
~ Alan Curtis Kay ~*

Introduction and Overview

1.1 Problem Description and Motivations

We are in an information era where generating, querying, extracting, and storing large amounts of digitized information are commonplace. A growing number of organizations and individuals daily and easily collect, but hardly handle, many terabytes and gigabytes of data respectively. The Web was originally conceived as a mean for sharing human-oriented documents. It was thought as a set of nodes in a network storing interlinked hypertext documents that can be rendered by browsers and presented to human users. But, today, the Web is the largest repository of human knowledge and acts as a worldwide collaborative platform [152]. Everyone can produce, share and improve a very diverse spectrum of (multimedia) content types in Social Networks, Deep Web Sites, Wikis, Blogs, etc. Hence, an enormous amount of information is now available through the Web. If computers can be enabled to extract and utilize the knowledge embedded in Web documents, a powerful knowledge resource for many application fields will be realized.

Sample Application Areas

There are a series of current and future applications that require the ability to describe, query, extract information available on the Web on the base of its semantics. The most common scenario is that users or companies want to extract data from external Web sources for which public Web Services, or other programmatic APIs, are not available. In fact, while numerous (often “Web 2.0”-oriented) data sources offer public APIs or structured data feeds such as RSS, many more sources do not. This is particularly true for high-value data extraction applications (such as the gathering of real-time competitive market intelligence). Several of real world applications that demonstrate the usefulness of information extraction follows.

Web Search Engine. Nowadays, Web keyword-based search, implemented in engines such as Google and Yahoo, is a very popular method to

search information on the web. Search engines only retrieve potentially interesting list of Web documents ranked according the term occurrences in the documents. However, in many cases users have specific needs that cannot be answered by an ordered set of documents. Users need objects (having a clear semantics) retrieved from different Web pages and arranged in a comprehensible way. Thus, Web search engines needs to recognize key entities (persons, locations, organizations, etc.) associated with a web page. For instance an user needs to booking an holer with specific characteristics, thus the aim is not obtain a list of Web pages, but obtain specific information.

Items comparison. The possibility for a user to compare different offerings of the same object is a feature currently not supported by all online e-commerce sites. To enable users to compare different offerings of the same object, records describing such object from different Web sites should be extracted.

Web market monitoring. Nowadays, much information about competitors such as information on profit, pricing, product availability, store locations, and so on, can be retrieved legally from public information sources on the Web, such as annual reports, public data bases, and employer's personal Web pages. These sources are principally in semi and unstructured formats. For this reason, online market intelligence (OMI), and especially competitive intelligence for product pricing, need Web data extraction.

For example, SOGEI (the ICT company of the Italian Ministry of Economy and Finance) would monitor property prices, by creating and constantly updating a knowledge base aimed at providing real-estate technical estimation survey services. In this case an object is represented by an apartment, where location, prices, facilities, etc. represent its properties. Such information should be automatically identified and extracted from Web pages of different real-estate agency sites, and stored in a knowledge base.

Other examples follows. A supermarket wishes monitor competitors product prices and immediately know special offers and new products, in order to be able to maintain its competitive position and offer better products. Online travel agency and booking hotel sites offering that guarantee best prices guarantee need to know offer over the Web by competing travel and booking hotel agencies. Moreover, they want to be informed about the average market price in order to maintain the maximum profit.

In all these and similar applications the relevant sources are a limited number and usually known. Such applications need Web data extraction technology with a very high level of data precision. So manual or semi-automatic wrapper generators for the relevant sources are preferable to generic Web harvesting tools.

Patent and official bulletins extraction. Public administration and companies need information extraction from patent and official bulletins of a public administration for industrial, commercial and public uses. For instance, could be necessary query a Web-based application, extract the set of result links, follow them, and extract the needed information on the result pages.

And also, could be necessary to automatically extract laws and calls related to european framework program, as well as their metadata, from the official bulletins.

Medical Information extraction. Another example scenario is the necessity to capture medical information from the Web as well as from clinical records. Acquired information could be used in clinical processes. The goal of the application could be extract semantic metadata about oncology therapies and errors with temporal data providing information to assist clinical decision, result analysis, error detection, etc. Extracted information could be analyzed for identifying main causes of medical errors, high costs and, potentially, suggesting clinical processes improvement able to enhance cost control and patient safety.

News Extraction. Information extraction can be applied to identify and aggregate interesting daily news articles from different Web news sites. For example, for the news articles about football matches, automatically provide a table of the players person names, the team names and the game results can be a very valuable result and save the time that a user spend in browsing.

Challenges

The complexity of the extraction process as well as the heterogeneous, unstructured and complicated nature of data and of document's internal structures pose interesting research challenges. More specifically, the relevant issues to be taken into account are: (i) the unstructured, semi-structured, and heterogeneous internal formats adopted for Web documents, (ii) their human oriented nature that implicates we have to manage natural language, spatial arrangement made available by presentation, and semantics of data and relations among them.

Two of the most diffused document formats are the Hypertext Markup Language (HTML) [182] and Adobe portable document format (PDF)[4]. HTML and PDF are semi and unstructured document formats conceived for presenting documents on screen or for printing respectively. In this dissertation we subsume documents encoded by HTML or PDF formats as *presentation-oriented documents* (PODs). The internal encoding of HTML documents is based on a tree structure, whereas PDF documents are described by a content stream, which contains a sequence of graphical and textual objects located at precise positions within the document pages that express only where contents have to be visualized after rendering. This makes it difficult to extract relevant content from such sources.

In the past, manual Web wrapper (that is a procedure that is designed to access HTML documents and export the relevant text to a structured format) construction (e.g. [164]), or Web wrapper induction approaches (e.g. [37, 150, 198]) have exploited regularities in the underlying document structures, which led to such similar layout, to translate such information into relational or logical structures. However, surveying a large number of real (Deep)

Web pages, we have observed that the document structure of current Web pages has become more complicated than ever implying a large conceptual gap between document structure and layout structure. Wrapping from PDF (and PostScript) has been recognized as a very important and challenging problem because PDF documents are completely unstructured and their internal encoding is completely visual-oriented. So traditional wrapping/information extraction systems cannot be applied. In [17] Gottlob et al. point out that “there is a substantial interest from industry in wrapping documents in format such as PDF and PostScript. In such documents, wrapping must be mainly guided by a reasoning process over white spaces... it is very different from Web wrapping”, hence, require new techniques and wrapping algorithms including concepts borrowed from the document understanding community. So, unfortunately, the internal structures of presentation-oriented documents, like DOMs, are often not convenient and sometimes even not expressive enough to allow for extracting the associated meaning, specially in the case of the PDF.

The spatial arrangement of objects obtained by visualizing HTML and PDF documents makes available visual cues which help human readers to make sense of document contents. A human reader is able to look at an arbitrary page of a POD and intuitively recognize its logical structure understanding the various layout conventions that have been used in the documents presentation, and s/he can also determine more complex relationships, for instance from data displayed in tabular form. In fact, Web designers plan web pages contents in order to provide visual patterns that help human readers to make sense of document contents. This aspect is particularly evident in *Deep Web* pages [122], where designers always arrange data records and data items with visual regularity or when tables are used to meet the reading habits of humans. Whereas web wrappers today dominantly focus on either the flat HTML code or the DOM tree representation of web pages, Wrapping from the visual representation of web pages can be particularly useful for layout-oriented data structures such as web tables and allows to create automatic and domain-independent wrappers which are robust against changes of the HTML code implementation. Moreover, much of the information stored in PODs is encoded in natural language, which makes it accessible to some people (which can read the particular language), but such information is less tractable by computer processing.

We aim at overcoming the syntactic-based processing of information available on the web. Current data/syntactic driven processing approaches (e.g. web search uses just keywords and not concepts) should move towards concepts/semantic driven processing where basic units of processing are concepts obtained by combining elementary atomic data. In order to allow for exploiting the semantic of information, ontologies that constitute a more abstract and semantically rich description of stored data are needed. In this way objects, human activities, and related information can be semantically tagged and provided services can be closer to the way in which humans are used to act and think. For reducing the burden of manual ontology creation au-

automatic and semi automatic approaches for learning ontologies can be used. In particular, research direction is oriented towards semi-automatic creation of domain ontologies because a completely automatic construction of good quality ontologies is in general not possible for theoretical, as well as practical reasons. Currently, several research projects in the areas of intranet search, community information management, and Web analytics, are already employing IE techniques to bring order to unstructured data. The common theme in all of these applications is the use of IE to process the input and produce a structured representation to support search, browsing, and mining applications [12, 44, 148, 162]. Web search engines are also employing IE techniques to recognize key entities (persons, locations, organizations, etc.) associated with a web page. This semantically richer understanding of the contents of a page is used to drive corresponding improvements to their search ranking and ad placement strategies.

In this dissertation innovative information extraction approaches, techniques, models and languages are provided.

1.2 Main Contributions

The present dissertation provides innovative approaches, methods, algorithms, and languages capable to exploit spatial and semantic features of presentation-oriented documents (PODs), i.e. Web pages and PDF documents, for querying and extracting information from them. Research activities carried out during my PhD have led to: (i) The definition of a unique spatial model for representing PODs having different internal formats. By such a model content items in documents having HTML and PDF internal encodings can be represented along with their spatial features and relations. (ii) The definition of techniques, based on heuristics and machine learning algorithms, for automatic recognition and extraction of tables from PODs. Automatic table extraction algorithms are very well performing as show by experiments. From the comparison with existing approaches results that defined algorithm constitute the current state of the art in this field. (iii) The definition of wrappers learning approaches based only on spatial PODs features. Defined techniques, in fact exploit only the spatial relations among content items in order to learn wrappers for repetitive records contained in Deep Web Pages. Such techniques do not require preprocessing of the HTML tags aimed at inferring spatial arrangement, at the contrary they use directly the layouted Web page. Experiments show that the SILA algorithm performs better than approaches available in literature. (iv) The definition of formal languages aimed at describing and querying PODs. In particular, the SXPath language allows user to write very intuitive pattern similar to XPath expressions and capable to exploit spatial features of PODs resulting in a more simple query mechanism for this kind of documents as experiments shown. The spatial grammars formalism allow user to write wrapper a set of spatial productions that can be intuitively de-

signed by considering only what the user see on the screen. Both XPath and spatial grammars open the doors to novel wrapper learning techniques that considers the spatial structure of PODs and the semantic of contents. (v) The definition of methods, languages and approaches for knowledge representation that support information extraction and semantic annotation by enabling the description and automatic recognition of concepts contained in PODs.

1.3 Outlook

This section provides the reader with an overview on the content of this thesis. Moreover a chapter dependency schema is sketched which allows to follow the path that motivated each individual chapter and understand how chapters are connected to each another.

1.3.1 Thesis' Structure

This thesis is organized in four parts. Part I provides, in Chapter 1, the description of addressed scientific and application problems, motivations and main contributions. In Chapter 2 background knowledge about presentation-oriented documents, Web query languages, formal languages and knowledge representation approaches and languages is presented to the reader.

Part II gives the state of the art about information extraction and Web querying. Chapter 3, discusses information extraction approaches for flat text documents. Chapter 4, describes methods and approaches for Web information extraction. In particular this chapter distinguishes among manual, semi-automatic and automatic approaches. Chapter 5 presents PDF information extraction approaches considering document understanding techniques, table recognition and extraction methods and recent approaches to PDF document wrapping. Chapter 6 provides an analysis of existing ontology based information extraction approaches. Finally, Chapter 7 query languages for Web and multimedia documents.

Part III gives the detailed description of results of research activities carried out during the PhD course and describes original contributions in the field of information extraction. In particular, Chapter 8 introduces the reference structure of a next generation information extraction system capable to exploit spatial and semantic features. Features of such a system have been made available by results described in next chapters. Chapter 9 presents the spatial documents models that allows for representing in unified way PODs having both HTML and PDF internal representation. Chapter 10 describes the PDF-TREX approach based on document understanding techniques that led to automatic table recognition and extraction, and the SILA approach for wrapper induction for Web documents. Techniques described in this chapter make use of spatial features of PODs made available in their spatial representations. Chapter 11 contains the presentation of the XPath language,

that extends XPath and allows for querying PODs by means of spatial navigation constructs, and the spatial grammars formalism allows for expressing wrapper as a set of productions of a CFG extended in order to consider spatial and semantic relations among elements displayed on PODs. Chapter 12 presents ontology-based information extraction capabilities made available by the XONTO approach. This approach exploits a particular version of spatial grammars that provide the ability to model and handle spatial relations, and the ability to consider domain knowledge about the information to extract modeled in a knowledge base. Chapter 13 describes visual capabilities made available by SILA, SXPath and PDF-TREX approaches. The chapter presents how such approaches make viable visual interfaces that allow to actually extract information from PODs by using only visual facilities. Chapter 14 provides the description of some applications of approaches described in previous chapters to real world scenarios.

Part IV concludes the dissertation by providing in Chapter 15 a discussion about obtained results and possible future work.

1.3.2 Reader's Guide

The present thesis has been written following a logical path interconnecting the various research contributions. Hence, it is possible to recognize in this work three main threads. The first is related to automatic information extraction from PODs. The second is related to PODs querying. The third is related to semantic information extraction from PODs approaches at providing HTML documents, which related to extraction information on Web documents. The second is related to extraction from PDF documents.

As for the first thread the reader interested in automatic approaches to information extraction from PODs can focus on Part I, Chapters 4 and 5 in Part II, Chapters 8, 9, 10, 13 and 14 in Part III, and on Part IV. A complete understanding of the second thread requires that the reader focuses on Part I, Chapters 7 in Part II, Chapters 8, 9, 11, 13, 14 in Part III, and on Part IV. Regarding the semantic information extraction thread the reader must consider Part I, Chapters 3 and 6 in Part II, Chapters 8, 9, 12, 13 and 14 in Part III, and Part IV.

The Figure 1.1 summarizes chapters organization and provides links between chapters in order to allow the reader to choice the part of the thesis in which s/he is interested.

1.3.3 Publications

Part of the material of this dissertation has been published in several journals, conferences and technical reports:

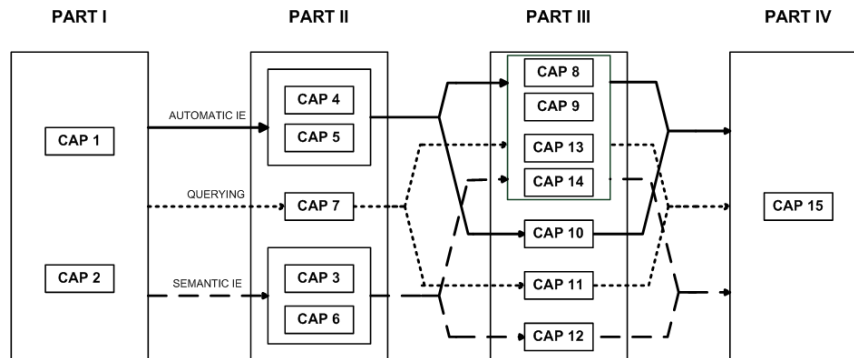


Fig. 1.1. Structure of the thesis and chapter dependencies

Journals

- **E. Oro**, M. Ruffolo, S. Staab. “*SXPath - Extending XPath towards Spatial Querying on Web Documents*” Journal Track VLDB - VLDB Endowment (PVLDB) Vol. 4, Issue 2, 129-140, 2010.
- **E. Oro**, M. Ruffolo, D. Saccà. “*Ontology-Based Information Extraction from PDF Documents with XONTO*” International Journal on Artificial Intelligence Tools (IJAIT), 2009.

Conferences

- **E. Oro**, F. Riccetti, M. Ruffolo. “*A Spatial Approach for Extracting Information from Presentation-Oriented Documents*”. International Conference on Agents and Artificial Intelligence (ICAART), 2011. To Appear.
- **E. Oro**, F. Riccetti, M. Ruffolo. “*ViQueL: A Spatial Query Language for Presentation-Oriented Documents*”. In: IEEE Int’l Conference on Tools with Artificial Intelligence (ICTAI), 2010. To Appear.
- **E. Oro**, M. Ruffolo. “*PDF-TREX: An Approach for Recognizing and Extracting Tables from PDF Documents*”. Tenth International Conference on Document Analysis and Recognition (ICDAR), 2009.
- **E. Oro**, M. Ruffolo, D. Saccà. “*Combining Attribute Grammars and Ontologies for Extracting Information from PDF Documents*”. 17th Italian Symposium on Advanced Database Systems (SEBD), 2009.
- **E. Oro**, M. Ruffolo, D. Saccà. “*A Semantic Clinical Knowledge Representation Framework for Effective Health Care Risk Management*”. 12th International Conference on Business Information Systems (BIS), 2009: 25-36.
- **E. Oro**, M. Ruffolo. “*Towards a Semantic System for managing clinical processes*”. In: 11th International Conference on Enterprise Information Systems (ICEIS), 2009.

- **E. Oro**, M. Ruffolo. “*XONTO: An Ontology-based System for Semantic Information Extraction from PDF Documents*”. In: IEEE Int’l Conference on Tools with Artificial Intelligence (ICTAI), 2008.
- **E. Oro**, M. Ruffolo. “*Towards a System for Ontology-Based Information Extraction from PDF Documents*”. In: Ontologies, DataBases, and Applications of Semantics (ODBASE), 2008.
- **E. Oro**, M. Ruffolo. “*Description Ontologies*”. In: International Conference on Digital Information Management (ICDIM), 2008.

Technical Reports

- **E. Oro**, M. Ruffolo, F. Valentini. “*SILA: A Spatial Instance Learning Approach from Deep Web Pages*”. ICAR-CNR Technical Report, 2010.

Background

This dissertation focuses on the definition of new approaches, methods, languages and tools for querying and extracting information from presentation-oriented documents, even using semantic approaches. In this chapter, first the Web and PDF documents are deeply analyzed showing their different internal structure, but that they all are conceived for presenting documents on screen or for printing. The spatial arrangement of objects obtained by visualizing presentation-oriented documents makes available visual cues which help human readers to make sense of document contents. Then, the widely accepted Web query languages and the formal languages are introduced. They will be the base used for defining innovative languages able to exploit the spatial information. Finally, knowledge representation will be introduced because it represents a fundamental ingredient that allows for recognizing concepts in textual documents for extraction and semantic annotation purposes.

2.1 Presentation-Oriented Documents – PODs

Documents are pervasive in human society, and there exist many internal formats and applications used for exchanging contents. The Web represents the biggest “document-sharing system”. Two of the most diffused document formats are the Hypertext Markup Language (HTML) [182] and Adobe portable document format (PDF)[4]. HTML and PDF are semi and unstructured documents conceived for presenting documents on screen or for printing respectively. A human reader is able to look at an arbitrary page and intuitively recognize its logical structure understanding the various layout conventions that have been used in the documents presentation and it can also determine more complex relationships for instance from data displayed in tabular form. In this dissertation we subsume documents encoded by HTML or PDF formats as *presentation-oriented documents*.

However, such document formats have a very different internal representation. The internal encoding of HTML documents is based on the tree structure,

whereas PDF documents are described by a content stream, which contains a sequence of graphical and textual objects located at precise positions within the document pages that express only where contents have to be visualized after rendering. This makes it difficult to extract relevant content from such various sources.

2.1.1 Internal Representations

Documents representation can be roughly divided in three types: tree-based, page-oriented, and stream-oriented. In the following we describe more in details their internal representation and their presentation features, in order to perceive which are the challenges for convert from unstructured or semi-structured format into structured and therefore machine-understandable format such as, for example, XML.

Tree Representation

By the mid-1980s, the diversity of incompatible markup languages and word-processing representations was making collaboration between authors quite difficult. In response, two competing document interchange formats were developed, the Standardized Generalized Markup Language (SGML) [95] and the Open Document Architecture (ODA). Only SGML was a success and its success was limited. However, SGML was the basis for the Hypertext Markup Language (HTML) used on the World Wide Web.

HTML is used as a page description language, HTML tags specify how the tagged information is to be presented on a browser. However, there are no semantics associated with HTML tags, nor are there semantics associated with the hyperlinks between documents, thus preventing automated interpretation of the contained information. So, the Web community attempt to find a high-level tree-structured specification. The solution was the Extensible Markup Language (XML) [27]. XML allows to represent semantics and to support applications, and it has become a standard representation for data interchange. However, remain the HTML is the actual document format used on the Web. HTML documents contain abundant hypertext markup information, both for indicating structure as well as for giving page rendering hints, next to informative textual content. HTML/XML are tree-structured and can have any level of complexity. HTML and XML documents in the following will be both called as XML documents for simplicity. An unranked ordered and labeled tree represent a Web document.

Definition 2.1. *Unranked ordered trees with node labels from a finite set of symbols Σ correspond closely to parsed XML documents. In an unranked tree, each node may have an arbitrary number of children. An unranked ordered tree can be considered as a structure of relational schema*

$$\tau = \langle V, \text{root}, (\text{label}_a)_{a \in \Sigma}, R_{\Downarrow}, R_{\Rightarrow} \rangle$$

where:

- V is the set of nodes in the tree.
- root is a unary relation, which contains the root of the tree.
- $R_{\Downarrow}, R_{\Rightarrow}$ are the firstchild and nextsibling binary relations respectively. $\text{firstchild}(\mathbf{n}_1, \mathbf{n}_2)$ is true if \mathbf{n}_2 is the leftmost child of \mathbf{n}_1 ; $\text{nextsibling}(\mathbf{n}_1, \mathbf{n}_2)$ is true if, for some i , \mathbf{n}_1 and \mathbf{n}_2 are the i^{th} and $(i+1)^{\text{th}}$ children of a common parent node, respectively, counting from the left. For instance, see Fig. 2.1.
- label is a unary relation, such that $\text{label}_a(\mathbf{n})$ is true if \mathbf{n} is labeled a in the tree.

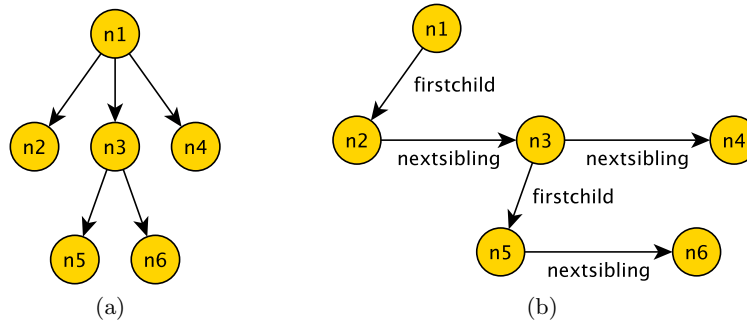


Fig. 2.1. (a) An unranked tree; (b) its representation using the binary relations “firstchild” and “nextsibling”.

Parsing of XML documents is the process of reading an XML document and providing an interface to the user application for accessing the document. A parser is a module that realizes these tasks and, in general, check if the document is *well-formed*.

There are mainly two categories of XML programming interfaces, DOM (*Document Object Model*) and SAX (*Simple API for XML*).

DOM is a tree-based interface that models an XML document as a tree of nodes (such as elements, attributes, texts, comments, etc.). A DOM parser maps an XML document into such a tree rooted at a Document node, upon which the application can search for nodes, read their information, and update the contents of the nodes. DOM was initially used for modeling HTML by various Web browsers. As inconsistencies existed among the individual DOM models adopted by different browsers, inter-operability problems arose in developing browser-neutral HTML codes. W3C (*World Wide Web Consortium*) standardized and released DOM Level 1 specification in 1998, with support for both HTML and XML. DOM Level 2 was released in 2000 and

added namespace support. The latest specification DOM Level 3 was released in 2004.

SAX is an event-driven interface. The application receives document information from the parser through a `ContentHandler` object. It implements various event handlers in the interface methods in `ContentHandler`, and registers the `ContentHandler` object with the SAX parser. The parser reads an XML document from the beginning to the end. When it encounters a node in the document, it generates an event that triggers the corresponding event handler for that node. The handler thus applies the application logic to process the node specifically. SAX was developed in 1997, in order to create a parser-independent interface. SAX1 was released in 1998, whereas, the latest release SAX2, in 2000 and it includes namespace support.

The SAX and DOM interfaces are quite different and have their respective advantages and disadvantages.

In general, DOM is convenient for random access to arbitrary places in an XML document, can not only read but also modify the document, although it may take a significant amount of memory space. To the contrary, SAX is appropriate for accessing local information, is much more memory efficient, but can only read XML.

- DOM is not memory efficient since it has to read the whole document and keep the entire document tree in memory. Therefore it is impossible to use DOM to process very large XML documents. However, HTML Documents are a “limited-size” version of markup languages, in fact they are generally aimed at representing single Web pages and not whole databases. In contrast, SAX is memory efficient since the application can only keep the portion that is of interests to the application, thus a SAX parser can easily handle very large documents.
- DOM is convenient for complex and random accesses that require global information of the XML document, whereas SAX is more suited for processing local information coming from nodes that are close to each other. The document tree provided by DOM contains the entire information of the document, therefore it allows the application to perform operations involving any part of the document. In comparison, SAX provides the document information to the application as a series of events. Therefore it is difficult for the application to handle global operations across the document. For such complex operations, the application would have to build its own data structure to store the document information. Consequently, the data structure may become very complex. In order to perform query that need consider not only local information, the DOM results more convenient.
- Since DOM maintains information of the entire document, its API allows the application to modify the document or create a new document, while SAX can only read a document.

- SAX is appropriate for streaming applications since the application can start processing from the beginning, while with DOM interface the application has to wait till the entire document tree is built before it can do anything.

Nowadays lots information available in Web sites is coded in form of HTML documents. The reasons are: the simplicity and power of HTML authoring tools, together with a valuable inertia to change markup language. Although HTML and XML are both derived from SGML, the former was designed as a presentation-oriented language, whereas XML separates data and its human-oriented presentation, which allows data-centered applications to better handle large amounts of data. A fundamental advantage of XML is the availability of powerful instruments for querying XML documents, namely XQuery/XPath [183]. In this thesis, the DOM approach is been adopted because the goal is parse HTML pages, allow for querying such presentation-oriented documents using also global conditions, and modifications on document may be performed. In Figures 2.2 and 2.3 are depicted a Web page taken from the `last.fm` social network related to music and a sketch of the DOM related to the part of the HTML document containing the profile of the Radiohead music band respectively. In Fig. 2.3 Grey circles and boxes are used for those nodes that are visualized on screen, whereas white circles and boxes depicts nodes that belong to the DOM but are not visible on screen.

Page-Oriented representation

The development of the laser printer required a means to transmit a page from computer to printer over the low-bandwidth connections then available. In response, various companies designed proprietary page description languages that described pages at a higher level, thus requiring substantially less bandwidth. The most important page-oriented document representation was Adobes PostScript, used in the first personal laser printers. Today the most important format is Adobes Portable Document Format (PDF) [4] because it is printer-independent, compact, and because Adobe distributes free viewing and printing software for all widely-used platforms. It is the de facto standard for print-oriented documents. In contrast, the output quality of HTML documents when printed (e.g. using a web browser) is not high enough for most professional applications. Perhaps this is because HTML has its roots in the scientific community, and not the publishing community. PDF have become increasingly popular for electronic publishing, mainly due to the preservation of the document layout across different platforms, widely used in enterprises and on the web, thought for print and on screen visualization. Business documents are usually represented in PDFs format and historical, legal, and financial documents are widely scanned and are represented as images, which with OCR techniques can be transformed in readable PDF. So, it can be considered the standard format for document publication, sharing and exchange.

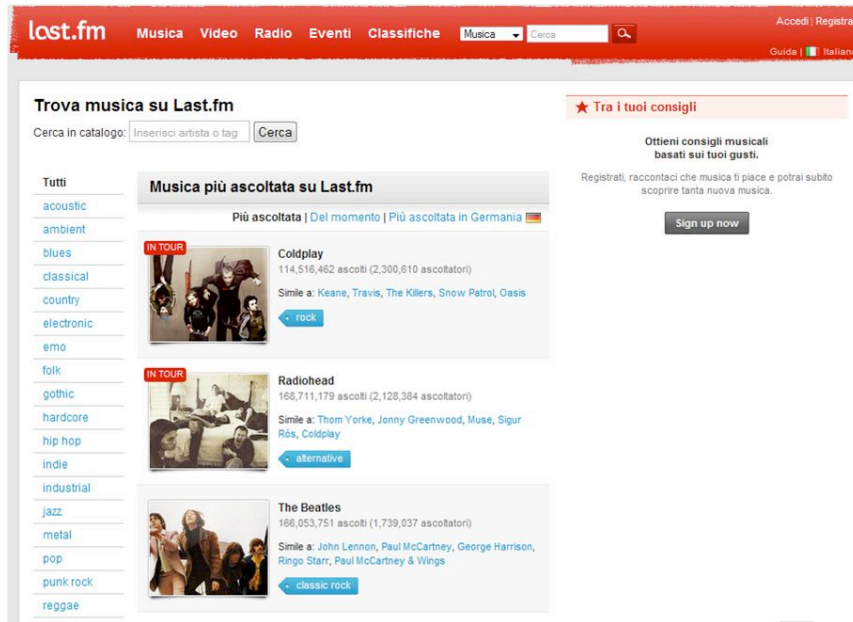


Fig. 2.2. A Page of the http://www.lastfm.it/ Web Site

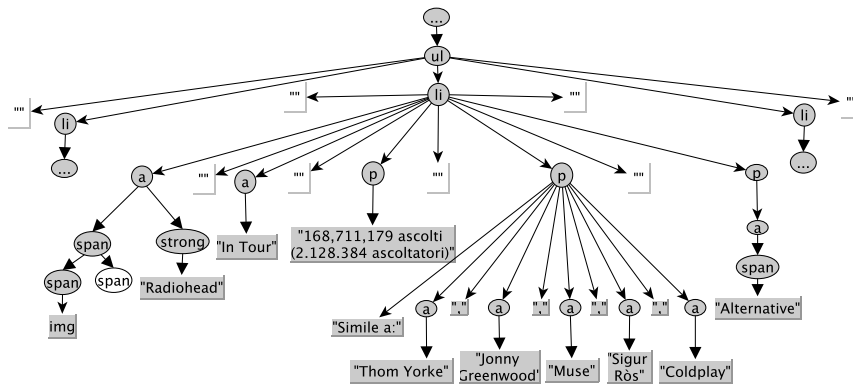


Fig. 2.3. A DOM Portion of the Web Page in Fig. 2.2

The content of a PDF is not organized in a structured way, since it lacks markups that express the document logical structure. In fact, PDF is a *document description language*, which describes a sequence of graphical and textual elements located at precise positions within the document pages. This feature guarantees that the visual rendering of a PDF document is independent from the PDF reader.

Each element holds some metadata about presentation features, and 2-dimensional coordinates that express the position in which it must be shown on the page at visualization or printing time. Elements can appear in casual order in the content stream, so the appearance of contents of a PDF document can be understood only after page rendering or printing. The order and the elements in the data stream depends on how content stream of the specific PDF document is generated. For instance the PDF document shown in Figure 2.4 represents a table, but its elements are presented in the stream in sparse order and often too small and do not represent logical units of contents as cells.

Electricity Production by Fuel									Change
Direct Energy Content [TJ]	1994	1996	1998	2000	2004	2005	2006	2007	'94 - '07
Total Gross Electricity Production	144 708	192 879	147 998	129 776	145 583	130 468	164 199	140 964	-2.6%
Oil	9 547	20 808	17 906	15 964	5 881	4 933	5 811	4 616	-51.7%
- Orimulsion	-	14 495	12 890	13 467	7	-	-	-	*
Natural Gas	8 206	20 442	29 260	31 589	35 807	31 606	33 903	24 886	203%
Coal	119 844	142 795	85 151	60 022	67 232	55 665	88 439	71 631	-40.2%
Surplus Heat	-	123	136	139	40	-	-	-	*
Waste, non-renewable	463	610	702	994	1 163	1 459	1 472	1 416	206%
Renewable Energy	6 647	8 101	14 844	21 068	35 459	36 805	34 574	38 415	478%
Solar Energy	0	1	1	4	7	8	8	9	3 017%
Wind Power	4 093	4 417	10 152	15 268	23 699	23 810	21 989	25 823	531%
Hydro Power	117	69	98	109	95	81	84	101	-14.1%
Biomass	2 116	3 207	3 911	4 936	10 646	11 889	11 517	11 504	444%
- Straw	293	748	960	654	3 057	3 088	3 359	3 185	988%
- Wood	429	340	512	828	3 546	3 730	3 041	3 398	691%
- Waste, renewable	1 393	2 120	2 439	3 454	4 043	5 071	5 117	4 921	253%
Biogas	321	407	682	751	1 013	1 017	976	978	205%

Fig. 2.4. A PDF Document about Tabular Data on Electricity Production

Whereas in HTML it is possible to parse the source code to find the locations of certain structures such as headings and tables, such machine-readable information is usually missing from PDF files. This logical structure information is critical for automatic data extraction, and is also useful for applications such as search, indexing and accessibility for disabled persons or mobile devices. Later versions of the PDF format have attempted to get around this limitation by providing support for tagging [99]. Tagged PDF extends the page description core of PDF with a structural tagging system to encode the roles of text fragments (e.g., body text, footnote, etc.), adds explicit word breaks, and maps all fonts to Unicode. Used properly, Tagged PDF ensures

that the content of a PDF document can be scanned in the same order that a human reader would scan it and clearly identifies elements like marginal notes and headers that are not part of the main text ow. It also supports search and indexing, as well as being able to encode some of the semantics of XML. However, the fact remains that the vast majority of PDF files either do not include such meta-information, or this information is not rich enough to enable us to locate the data instances to be extracted. The process by which most PDFs are created is still based on printing the document to a virtual printer driver, which strips the document of all its meta-information. Any logical information must then be added manually after the PDF file has been created.

Stream-Oriented Representation

Stream-oriented representations is used to represent flat textual information. Documents are seen as a sequence of characters or paragraphs. They may contain substantial amounts of formatting information, but unlike the page-oriented representations, generally do not encode the exact appearance of the document on the page or screen. The principal stream-oriented representations are raw text, the Rich Text Format (RTF), and various word processor formats. A raw text document contains a sequence of characters. Any organization of the characters into lines, paragraphs, or pages is specified by the use of specialized characters such as the ASCII line feed and form feed characters. The most common character coding schemes are ASCII and Unicode formats. Raw text has the advantages of simplicity, compactness, portability, and ease of processing. Its primary disadvantage is the inability to represent almost any useful typographic, hypertext, or multimedia effect. The raw text representation is remarkably robust and remains in widespread use, especially in the software development community, where the ubiquity of programming tools makes raw text an attractive representation. It is also a common representation for e-mail. RTF [100] and Word processor [134] representations describe a document as a sequence of paragraphs.

2.1.2 Displaying PODs

Presentation-oriented documents are visualized by layout engines embedded in web browsers (e.g. Netscape Gecko for Firefox¹) and PDF visualizers (e.g. Adobe Reader²). The main feature of layout engines and PDF visualizers is that they consider the area of the screen aimed at visualizing a document, as a 2-dimensional Cartesian plane on which they arrange *document content items* (i.e. images, alphanumeric strings, graphical and typographical elements of various kind). More in detail, layout engines of web browsers exploit the

¹ <https://developer.mozilla.org/en/Gecko>

² <http://www.adobe.com/products/reader/>

internal representation of web documents in terms of document object model (DOM) [181] in which content items correspond to the leaf nodes. Layout engines assign to each node in the DOM a *visualization area* on the plane computed by applying rendering rules. Rendering rules take into account the DOM structure and *cascade style sheets* (CSS) [180] that equip the web document. Likewise, PDF visualizers interpret parameters encoded in the PDF stream [4] and assign a visualization area on the plane to each content item.

By considering the web page shown in Figure 2.2 and the PDF document shown in Figure 2.4, visualization areas assigned, exploiting the layout engine Gecko and the Adobe Reader, to DOM nodes and content items are depicted in Figure 2.5 and 2.6 respectively. As shown in the figures visualization areas are rectangles having sides parallel to the axes of the cartesian plane.

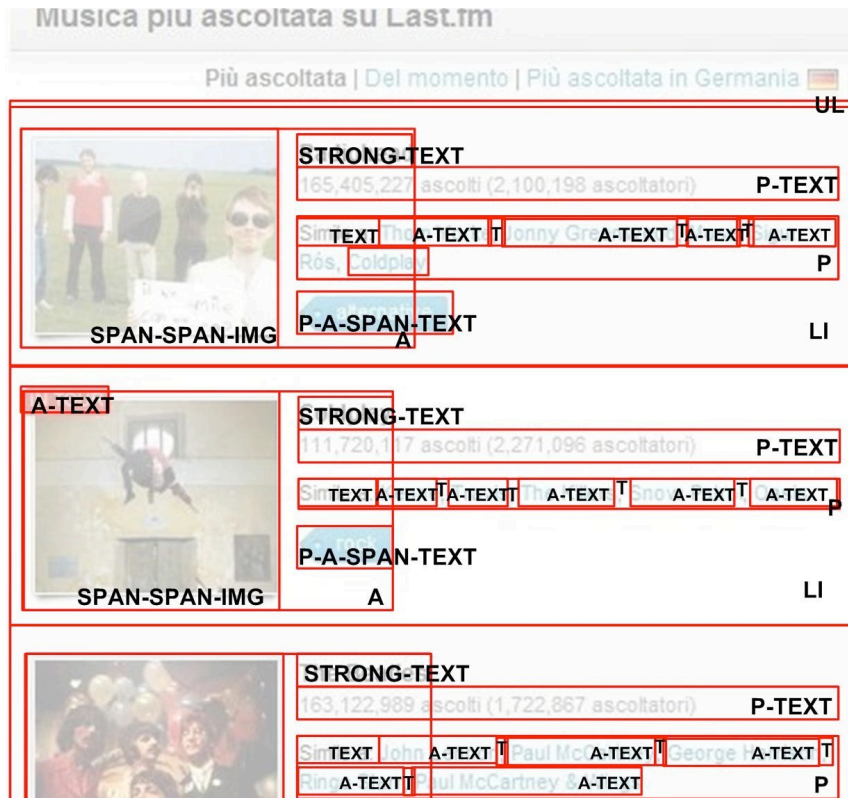


Fig. 2.5. Rectangles that Bound Visualized DOM Nodes

Electricity Production by Fuel									
Direct Energy Content (TJ)	1994	1996	1998	2000	2004	2005	2006	2007	Change '94 - '07
Total Gross Electricity Production	144 708	192 879	147 998	129 776	145 583	130 468	164 199	140 964	-2.6%
Oil	9 547	20 808	17 906	15 964	5 881	4 933	5 811	4 616	-51.7%
- Orimulsion	-	14 495	12 890	13 467	7	-	-	-	-
Natural Gas	8 206	20 442	29 260	31 589	35 807	31 606	33 903	24 886	203%
Coal	119 844	142 795	85 151	60 022	57 232	55 665	88 439	71 631	-40.2%
Surplus Heat	-	123	136	139	40	-	-	-	-
Waste, non-renewable	463	610	702	994	1 163	1 459	1 472	1 416	206%
Renewable Energy	6 647	8 101	14 844	21 068	35 459	36 805	34 574	38 415	478%
Solar Energy	6	8	1	2	7	8	8	6	8 017%
Wind Power	4 093	4 417	10 152	15 268	23 699	23 810	21 989	25 821	531%
Hydro Power	117	69	98	109	95	81	84	101	-14.1%
Biomass	2 116	3 207	3 911	4 936	10 646	11 889	11 517	11 504	444%
- Straw	293	748	960	554	3 057	3 088	3 359	3 185	988%
- Wood	423	340	512	828	3 546	3 730	3 041	3 398	691%
- Waste, renewable	1 393	2 120	2 439	3 454	4 043	5 071	5 117	4 921	253%
Bioogas	321	407	682	751	1 013	1 017	976	976	205%

Fig. 2.6. Rectangles that Bound Content Items in PDF Document

2.1.3 Challenges

The spatial arrangement of objects obtained by visualizing *presentation-oriented documents* makes available visual cues which help human readers to make sense of document contents. In fact, Web designers plan web pages contents in order to provide visual patterns that help human readers to make sense of document contents. This aspect is particularly evident in *Deep Web* pages [122], where designers always arrange data records and data items with visual regularity or when tables are used to meet the reading habits of humans. In Fig. 2.2 we show a deep web page coming from the last.fm social network. For instance, information about the two bands ‘Coldplay’ and ‘Radiohead’ in Fig. 2.2 is given using similar layout.

In the past, manual Web wrapper (that is a procedure that is designed to access HTML documents and export the relevant text to a structured format) construction (e.g. [164]), or Web wrapper induction approaches (e.g. [37, 150, 198]) have exploited regularities in the underlying document structures, which led to such similar layout, to translate such information into relational or logical structures. However, surveying a large number of real (Deep) Web pages, we have observed that the document structure of current Web pages has become more complicated than ever implying a large conceptual gap between document structure and layout structure. Wrapping from PDF (and PostScript) has been recognized as a very important and challenging problem because PDF documents are completely unstructured and their internal encoding is completely visual-oriented. So traditional wrapping/information extraction systems cannot be applied. There is a substantial interest from industry in wrapping documents in formats such as PDF and PostScript. Wrap-

ping of such documents must be mainly guided by a visual reasoning process over white space and Gestalt theory, which is substantially different from web wrapping and, hence, requires new techniques and wrapping algorithms including concepts borrowed from the document understanding community.

Unfortunately, the internal structures of Presentation-oriented documents, like DOMs, are often not convenient and sometimes even not expressive enough to allow for extracting the associated meaning, specially in the case of the PDF. Typical problems are incurred by the separation of document structure and the ensued spatial layout — whereby the layout often indicates the semantics of data items. E.g. the meaning of a table cell entry is most easily defined by the leftmost cell of the same row and the topmost cell of the same column (in Western languages). In tree structures such as arising on real-world web pages, such spatial arrangements are rarely explicit and frequently hidden in complex nestings of layout elements — corresponding to intricate tree structures that are conceptually difficult to query. Even if they offer fine-grained annotation the conceptual gap between the low level DOM representation and the semantics of the elements is very wide making the querying task very hard for a human querying the document or for a machine aiming at the automated learning of extraction rules. Thus, it has become very difficult: (i) for human and applications aiming at manipulating web contents (e.g. [75, 102, 164]), to query the web by language such as XPath 1.0; (ii) for existing wrapper induction approaches (e.g. [150, 198]) to infer the regularity of the structure of deep web pages by only analyzing the tag structure. Hence, the effectiveness of manual and automated wrapper construction are limited by the requirement to analyse HTML documents with increasing structural complexity whereas the intrinsic print/visual oriented nature of PDF encoding poses many issues in defining “ad hoc” IE approaches. Moreover, existing approaches are not able to generate extraction rules reusable when the internal structure changes and for different Web sites where information is presented by the same visual pattern. Nevertheless, two very needed property of wrappers are resilience and adaptiveness. Resilience is the capacity of continuing to work properly in the occurrence of changes in the pages to which they are targeted. Adaptiveness is the desirable capability that a wrapper built for pages of a specific Web source on a given application domain could work properly with pages from another source in the same application domain.

Thus, the exploitation of the only internal structure is not enough. A solution is allow for exploiting also visual representation available for all presentation-oriented documents. For instance, a human reader can relate information on the page in Fig. 2.2 by considering the spatial arrangement of laid out content elements. He can interpret the spatial proximity of images and nearby strings as a corresponding aggregation of information, namely as the complete record describing the details of a music band profile and one of its photos. Thus, details of a music band profile can be described in the following “qualitative spatial” way: *a music band profile is: the music band photo that has at east its descriptive information. That is its name, a number*

of hits, a list of similar music artists and a music genre, which are on top of each other. By considering Figure 2.6 the particular spatial arrangement of content items in the PDF document suggests to a human reader that he/she is looking at information organized in tabular form. The meaning of each content item in the table comes from the alignment of rows and columns. So the reader interprets strings in the first row and first column as row and column headers and each number in the table body on the base of its headers. By exploiting spatial information it is possible to acquire table data and store them in structured form. As shown, this method can be particularly useful for layout-oriented data structures and allows to create automatic and domain-independent wrappers which are robust against changes of the HTML code implementation, and also applicable to other presentation oriented formats.

2.2 Web Query Languages

World Wide Web consortium (W3C) is an international consortium for development of World Wide Web protocols and guideline aimed at facilitating interoperability and avoid fragmentation of the Web. It was founded in 1994 by the inventor of the World Wide Web Tim Berners-Lee. The consortium consists of member organization and dedicated staff of technical experts. Membership is open to any organization or individual whose application is reviewed and approved by the W3C. W3C is responsible for such technologies as HTML, XHTML, XML, CSS, XPath, XQuery, etc.

In this section the languages for querying XML documents will be described.

2.2.1 XPath

XPath (XML Path Language) is a simple and easy-to-use query language proposed by W3C for addressing portions of XML documents.

XPath was designed in order to address the need of a language for matching pattern in XML documents. The W3C XML Linking working group, which was working on XLink and XPointer, decided to develop a common language. Thus the first version XPath 1.0 was published in 1999 [184] XPath allows to navigate and select nodes in the DOM of web documents (i.e well formed HTML documents). The language gets its name from the use of a path notation. In addition it also allows some minor computations resulting in values such as strings, numbers or booleans. The semantics of the language is based on a representation of the information content of an XML document as an ordered tree (see Definition 2.1). The set V contains seven types of nodes: *root*, *element*, *text*, *attribute*, *namespace*, *processing instruction*, and *comment nodes*. Three properties can be associated with these nodes: a local name, a namespace name and a string-value. The local name is defined for element, attribute, processing instruction and namespace nodes. The string-value is

defined for all types of nodes, but for the root and element node it is derived from the string-values of its children nodes.

Over the nodes in this tree a document total order $<_{doc}$ is defined.

Definition 2.2. *The document total order $<_{doc}$, orders the nodes as they are encountered in a pre-order walk of the tree. Let $u, w \in V$ be two nodes, then $u <_{doc} w$ iff the opening tag of u precedes the opening tag of v in the (well-formed version of the) document.*

An XPath expression consist usually of a series of steps that each navigate through this tree in a certain direction and select the nodes in that direction that satisfy certain properties. The primary syntactic construct in XPath is the *expression*. The evaluation of an expression yields a result which has one of the four basic types: *node set*, *Boolean*, *number* or *string*. The most important kind of expression is the *location path*. Location paths allow web document navigation and recursively contain expressions that are used to filter sets of nodes. Location paths are sequences of *location steps* separated by the navigation operator $/$.

Definition 2.3. *The formal syntax of a location step is:*

$$\chi :: t[p_1] \dots [p_n]$$

where:

- χ is an axis name. Axis names are *self*, *child*, *parent*, *descendant*, *descendant-or-self*, *ancestor*, *ancestor-or-self*, *next-sibling*, *following-sibling*, *previous-sibling*, *preceding-sibling*, *following*, and *preceding*. Axes *ancestor* and *descendant* are the transitive closure of *parent* and *child* respectively, while axes *ancestor-or-self* and *descendant-or-self* are the reflexive and transitive closure of *child* and *parent* respectively. An axis name represents a relation $\chi \subseteq V \times V$ and has the intuitive meaning conveyed by the name itself. The axis selects a set of nodes that satisfy the relation. Axes are computed as functions $\chi : 2^V \rightarrow 2^V$ (overload the relation name) such that $\chi(V_c) = \{n \mid \exists n_c \in V_c : n_c \chi n\}$ where $V_c \subseteq V$ is a subset of nodes.
- t is a node test, that specifies the node type or the tag name of the nodes returned by a location step. The wildcard "*" matches all types and all labels.
- p_1, \dots, p_n are optional predicates which filter nodes returned by a location step. In predicates one may use: (i) Boolean, arithmetic, and comparison expressions based on related operators; (ii) functions that manipulate strings, numbers, and node positions w.r.t. the document total order. \square

A location step evaluation occurs with respect to a *context* and selects the node set that results from filtering, by means of the *node test* and *predicates*, the initial set of nodes returned by the *axis*. Note that the context refers to the tree nodes considered as current nodes for the evaluation of any XPath expression. Node sets, returned by location paths, can

be merged by using the union operator $|$. For instance, the simple location path: $/\text{descendant}::\text{a}/\text{child}::\text{b}[1] | / \text{child}::\text{b}$ starts from the root node $/$ and selects the set of direct children labeled b or the first (predicates $[1]$) children that are labeled with b of all the nodes that are labeled with a . XPath axes have an abbreviated syntax that allow for writing more concise location paths [184].

The tree structure, as well described in Gottlob papers [77], can be represented by means of *firstchild* and *nextsibling* functions defined in $V \rightarrow V$. *firstchild* returns the first child of a node (null if the node is a leaf) and *nextsibling* returns the right sibling, child of the same parent, (null if the node is the last child of its parent). The related binary relations are defined as $\{\langle u, f(u) \rangle | u \in V \wedge f(u) \neq \text{null}\}$, where $f \in \{\text{firstchild}, \text{nextsibling}\}$.

Traditional XPath axes are defined in terms of their *primitive* relations (i.e. *firstchild*, *nextsibling* and their inverses), as shown in Table 2.1 [76, 77]. $T_1.A_2$, $T_1 \cup T_2$ and T_1^* denote the concatenation, union, and reflexive and transitive closure, respectively, of binary relations T_1 and T_2 (primitive relations, or traditional axis relations). Let $E(\chi)$ denote the regular expression defining χ in table 2.1. It is important to observe that some axes are defined in terms of other axes, but that these definitions are acyclic/ non-recursive.

child	:= firstchild.nextsibling*
parent	:= (nextsibling ⁻¹)*.firstchild ⁻¹
descendant	:= firstchild.(firstchild \cup nextsibling)*
ancestor	:= (firstchild ⁻¹ \cup nextsibling ⁻¹)*.firstchild ⁻¹
descendant-or-self	:= descendant \cup self
following	:= ancestor-or-self.nextsibling.nextsibling* . descendant-or-self
preceding	:= ancestor-or-self.nextsibling ⁻¹ .(nextsibling ⁻¹)* .descendant-or-self
following-sibling	:= nextsibling.nextsibling*
preceding-sibling	:= (nextsibling ⁻¹)* nextsibling ⁻¹

Table 2.1. Traditional XPath Axes definition

In order to describe how node sets result from a location step using axis relation χ , the corresponding axis-function (and its inverse) that overloads axes relation names χ is defined.

Definition 2.4. Let χ denote an axis. The axis function, which overload the relation name χ , $\chi : 2^V \rightarrow 2^V$ is defined as $\chi(\Gamma) := \{u | \exists c \in \Gamma : c\chi u\}$, where $\Gamma \subseteq V$ is a set of nodes. Moreover, the inverse spatial axis function $\chi^{-1} : 2^V \rightarrow 2^V$ is defined $\chi^{-1}(\Gamma) := \{c \in V | \chi(\{c\}) \cap \Gamma \neq \emptyset\}$. \square

where $\Gamma \subseteq V$ is a set of nodes; Γ' is represented as a list*; $E(\chi)$ denote the regular expression defining $\chi \in \Delta$ [76, 77] in table 2.1; e_1 and e_2 are sub-regular expressions shown in table 2.1, T, T_1, \dots, T_n are primitive relations used to represent traditional axes, χ, χ_1 and χ_2 are axes.

As shown in [76, 77] let $\Gamma \subseteq V$ be a set of nodes of an XML document and χ be an axis. Then, $\chi(\Gamma) = \text{eval}_\chi(\Gamma)$ and the Algorithm 1 runs in time

Algorithm 1: Traditional Axis Evaluation

Input: A set of nodes Γ and an axis $\chi \in \Delta \cup \Delta_s$ Output: $\chi(\Gamma)$ Method: $eval_\chi(\Gamma)$

```

function  $eval_{self}(\Gamma) := \Gamma$ .
function  $eval_{\chi_t}(\Gamma) := eval_{E(\chi_t)}(\Gamma)$ .
function  $eval_{e_1.e_2}(\Gamma) := eval_{e_2}(eval_{e_1}(\Gamma))$ .
function  $eval_T(\Gamma) := \{T(u) \mid u \in \Gamma\}$ .
function  $eval_{\chi_1 \cup \chi_2}(\Gamma) := eval_{\chi_1}(\Gamma) \cup eval_{\chi_2}(\Gamma)$ .
function  $eval(T_1 \cup \dots \cup T_n) * (\Gamma)$  begin
   $\Gamma' := \Gamma$ ;
  while there is a next element  $u \in \Gamma'$  do
     $\Gamma' := \Gamma' \cup \{T_i(u) \mid 1 \leq i \leq n \wedge T_i(u) \neq null \wedge T_i(u) \notin \Gamma'\}$ ;
  return  $\Gamma'$ ;
end;

```

$O(|V|)$. The time bound is due to the fact that each of the eval functions can be implemented so as to visit each node at most once and there are $O(|V|)$ edges.

In last years, intuitive navigational features and querying capabilities of XML documents have made the XPath language central to most XML-related technologies and attracted much attention in the computer science research community. In particular, expressiveness and computational complexity of relevant XPath fragments have been studied in detail. For instance, Gottlob et al. in [77, 78] propose a complete and concise formal semantics of XPath 1.0 and a characterization of complexity classes of different fragments of this language. However, XPath poses some issues in navigating and querying the tree structure of presentation-oriented web documents. For instance, presented information could be arranged in intricate tree structures and do not represent valuable visual features and spatial relations. So to use XPath, as it is, could become very difficult. Furthermore, presentation-oriented document formats like the Adobe PDF cannot be addressed by XPath as it is. In fact the meaning of content items strongly depends on their spatial arrangement.

XPath 2.0. The current version of XPath is XPath 2.0. Its recommendation was published in 2007 [24]. The data model is largely the same as for XPath 1.0. The main change is that the root of a document is represented by a document node. Another important change is that the data structure of results is not a set of nodes but an ordered list of nodes and atomic values as defined by XML Schema. Do not exist nested lists, so (('a'), (('b'), ('c'))) is equivalent to (('a'), ('b'), ('c')). The concatenation of sequences s1

and s_2 is denoted as s_1, s_2 . XPath 2.0 extends used constructs of XPath 1.0 and it is semantically and syntactically a subset of XQuery 1.0.

2.2.2 XQuery

XQuery is a shorthand for XML Query Language. It has been a W3C recommendation since 2007 [25]. XQuery is query language for querying collections of XML documents (Web documents, contained in a database or in a file system) and combine the results into completely new XML fragments. It is a declarative, statically typed and it is based on XPath that represents its core. Its predecessor is Quilt [36]. From XML-QL [49] came the idea of using variable bindings in the construction of new values, and from SQL, the expressions **SELECT-FROM-WHERE** were taken. Originally, it was designed as a query language to query and transform XML data natively. By now, it is a functional programming language, and it is Turing complete. XQuery has evolved to a general purpose programming language that can read, update, and transform any kind of data besides XML. For instance, in [68, 69] XQuery is used to improve the programmability of Web browsers by enabling the execution of XQuery programs in the browser. Although it has the potential to ideally replace JavaScript, it is possible to run it in addition to JavaScript for more flexibility. In fact, XQuery allows DOM navigation and manipulation needed for programming the browser.

is based on the same data model as XPath 2.0.

Path expression defined in XPath 1.0 are almost all included and have mostly the same semantics, but they do not return a set of nodes but a sequence of nodes sorted in document order. More operators have been introduced, such as **intersect** and **difference** that correspond to the set intersection and set difference.

The access to collections of XML documents is provided by the functions `doc()` and `collection()` that both expect as argument a string containing a URI. These functions retrieve the requested XML fragments associated with this URI and, if this was not already done before, construct their data models and return a document node or a sequence of nodes that are the roots of the fragments.

XQuery has a broad functionality, covering simple expressions such as constants, variables, and comparisons to complex expressions for database queries, transformations, and information retrieval. The core expressions of XQuery are the **FLWOR** (pronounced “ower”) expressions corresponds to the **SELECT FROM WHERE** statement in SQL, which are illustrated by the following example:

```
for $x at $i in doc("book.xml")
let $tit := $x/tit
where $x/author ftcontains "kinsella"
return
```

```

<li>
  {$x/author}
  <title>{data($tit)}</title>
</li>

```

A FLWOR expression starts with one or more **for** and **let** clauses that each bind one or more variables (that always start with \$). The **for** clause binds variables such that they iterate over the elements of the result sequence of an expression, and the **let** clause binds the variable to the entire sequence. This is followed by an optional **where** clause with a selection condition, an optional **order by** clause that specifies a list of sorting criteria and a **return** clause that contains an expression that constructs the result.

An XQuery expression is evaluated in a context. The context contains functions, namespaces, schemas, and variable bindings. For instance, the expression **\$x** will be evaluated using the context; if the variable **\$x** is not defined in the context, then an error is raised as part of the evaluation of this expression. Otherwise, this expression is evaluated to the value of **\$x** as defined in the context. Likewise, function invocations are evaluated according to the definition of the functions in the context.

The W3C continue to work on several extensions of XQuery and XPath. A new extension for XQuery and XPath (XQuery 1.1 and XPath 2.1) is planned. The proposed extensions for XQuery include grouping on values, grouping on position, calling external functions and finally higher order functions. Moreover, in order to support search text, XQuery also involves full-text search [54] described in the following.

XQuery Full-Text

The XQuery and XPath languages are a powerful languages provide sophisticated structured query capabilities, they only provide rudimentary capabilities for querying the text (unstructured) parts of XML documents that are sufficient for simple sub-string matching but does not provide more complex search capabilities.

To address these short-comings, W3C has formulated the *XQuery 1.0 and XPath 2.0 Full-Text 1.0 Requirements* [53]. These requirements specify a number of features that must, should or may be supported by full-text search extensions to the XQuery and XPath languages. These extension add constructs for doing full text searches on selections of documents, text-search scoring variables that can be used in FLWOR expressions, and full-text matching options that can be defined in the query prolog.

XQuery Full-Text provides a full range of query primitives (also known as selections) that facilitate the search within the textual content of XML documents. The textual content is tokenized as a series of tokens which are the basic units to be searched. Intuitively, a *token* is a character, n-gram, or sequence of characters. An ordered sequence of tokens that should occur in the document together is referred to as a *phrase*.

XQuery Full-Text proposes four major features for support of full-text search the XQuery and XPath query languages:

- A XQuery/XPath expression returns a list of nodes, and after the keyword `ftcontains` a full-text selection can be specified.
- Primitives for supporting complex full-text search queries, such as token and phrase search, token ordering, token proximity, token scope, match cardinality, and Boolean combinations of the previous. Further, XQuery Full-Text allows control over the natural language used in the queried documents, the letter case in matched tokens, and the use of diacritics, stemming, thesauri, stop words, and regular-expression wildcards through the use of match options.
- the Data Model used for standard XQuery and XPath is inadequate for text management [29], in fact it represents data only about entire XML nodes. But more precise data model is needed [54] for representing information like the positions of the tokens within a node.
- Support for relevance scoring.

In the following a simple example of Full-text XQuery is shown:

```
for $b in /books/book
where $b/title ftcontains ("dog" with stemming) ftand "cat"
return $b/author
```

This example finds all authors of all books the title of which contains the word “cat”, as well as the word “dog” or a word with the same stem.

2.3 Formal Languages

Most existing approaches in order to extract information from unstructured document use regular grammars as a convenient mean for *extracting information from text automatically*. Indeed, regular grammars, offer a simple and declarative way to specify the patterns to be extracted, and are suitable for efficient evaluation (recognizing membership of a string to a regular language is linear-time doable). However, regular grammars have a *limited expressiveness*, which is not sufficient for a powerful information extraction. There are simple extraction patterns, like, for instance, $a^n b^n$, that are relevant for information extraction but cannot be expressed by a regular grammar. To express patterns of this kind more powerful grammars have to be considered. For instance, in order to express natural language sentences it is necessary to make use of context-free grammars. Moreover, because the grammar complexity grows, the study of efficient parsing techniques is essential.

In the following, grammars formalisms and parsing techniques used in this dissertation are described.

2.3.1 Grammars

A formal grammar is a set of rules of a specific kind, for yielding strings in a formal language. The rules describe how to form strings from the language's alphabet, that are valid according to the language's syntax.

Context-Free Grammars

A *context-free grammar* $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ consists of an alphabet of *terminals* Σ , an alphabet of *nonterminals* N , a *start symbol* $S \in N$, and a finite set of *productions* $\Pi \subseteq N \times (\Sigma \cup N)^*$. A production $p \in \Pi$ is said to be an ε -production if $p \in (N \times \{\varepsilon\})$. We usually write $A \rightarrow \alpha$ for the pair $\langle A, \alpha \rangle \in \Pi$. Let $\Rightarrow_{\mathcal{G}}^+$ be the binary relation of *derivation* w.r.t. \mathcal{G} , the language generated by \mathcal{G} is defined as $\mathcal{L}(\mathcal{G}) = \{x \in \Sigma^* : S \Rightarrow_{\mathcal{G}}^+ x\}$. A context-free grammar \mathcal{G} is *cycle-free* (or *non-cyclic*) if there is no derivation of the form $A \Rightarrow_{\mathcal{G}}^+ A$ for some nonterminal A .

We will often consider a subclass of context-free grammars called the *Chomsky Normal Form*. A grammar G is an element of *CNF* iff all $p \in \Pi$ are either of the form $A \rightarrow a$ or $A \rightarrow BC$. Every context-free grammar G for which $\varepsilon \notin L(G)$ can be rewritten into an equivalent grammar in *CNF*.

Given a context-free grammar $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$, a *parse tree* t of \mathcal{G} is a tree where: (i) the *root node* $\rho(t)$ is labeled with the start symbol S ; (ii) each *leaf node* has label in $\Sigma \cup \{\varepsilon\}$; (iii) each *internal node* is labeled with a nonterminal; (iv) if $A \in N$ is a non-leaf node in t with children $\alpha_1, \dots, \alpha_h$ taken from left to right, then $A \rightarrow \alpha_1 \dots \alpha_h$ is a production in Π . By concatenating the leaves of t from left to right we obtain the derived string $x(t)$ of terminals, which is called the *yield* of the parse tree. The language generated by \mathcal{G} can be also defined as $\mathcal{L}(\mathcal{G}) = \{x(t) : t \text{ is a parse tree of } \mathcal{G}\}$. As defined in [127] the *size* of a parse tree is the number of its non-leaf nodes.

A context-free grammar \mathcal{G} is unambiguous if it does not have two different parse trees for any string. A context-free language is unambiguous if it can be generated by an unambiguous context-free grammar. Context-free grammars and languages are ambiguous if they are not unambiguous. Moreover as shown in [188] for cycle-free context-free grammars the ambiguity function is either an element of $2^{\Theta(n)}$ or of $\mathcal{O}(n^{\Theta(1)})$.

As defined in [119], a context-free grammar is said to be *deterministic* (*DCFG*), if it can be implemented (parsed) by a *deterministic pushdown automaton*. The deterministic context-free languages **DCFL** are a proper subset of the set of languages that possess an unambiguous context free grammar.

Some examples on the main three classes are given:

- $L_1 = \{a^n b^n c^n : n > 0\} \in \mathbf{CSL}$. No *context-free* grammar there exists for L_1 .
- $L_2 = \{a^n b^n : n > 0\} \in \mathbf{DCFL}$. An unambiguous *context-free* grammar for L_2 is: $S \rightarrow aA$, $A \rightarrow Sb|b$. No *regular* grammar there exists for L_2 .

A context-free grammar $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ is *linear* if $\Pi \subseteq N \times (\Sigma^* V \Sigma^* \cup \Sigma^*)$.

As defined in [89], a *deterministic linear grammar* $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ is a linear grammar where the two following conditions hold: (i) all production rules are of the form $A \rightarrow \varepsilon$ or $A \rightarrow aBw$ with $A, B \in N$, $a \in \Sigma$, and $w \in \Sigma^*$; (ii) $A \rightarrow a\beta$

Context-free grammars (CFGs) can themselves be divided into several *complexity* categories:

$$LL(k) \subseteq LR(1) \subseteq Det-CFG \subseteq Unamb-CFG \subseteq CFG$$

Regular Grammars.

A *regular grammar* is a context-free grammar $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ where either $\Pi \subseteq N \times (\Sigma \cup (\Sigma \circ N) \cup \{\varepsilon\})$ (right-regular grammar) or $\Pi \subseteq N \times (\Sigma \cup (N \circ \Sigma) \cup \{\varepsilon\})$ (left-regular grammar) holds. By construction any regular grammar \mathcal{G} is non-cyclic, so the size of any parse tree t of \mathcal{G} is $|x(t)|$.

Example 2.5. $L_3 = \{a^n b : n \geq 0\} \in \mathbf{REG}$. A regular grammar for L_3 is:

$$S \rightarrow aS|b.$$

Regular grammars can also be ambiguous. Let us show the grammar \mathcal{G} having the following productions: $S \rightarrow aS|aA|\varepsilon$, $A \rightarrow aS|aA|\varepsilon$. It is easy to see that $\mathcal{L}(\mathcal{G}) = \{a^n : n \geq 0\}$ but each string w of length n can be derived by exactly 2^n different parse trees. As shown in [170] “*every regular language is unambiguous*”. This means that for every regular language L there exists an unambiguous regular grammar \mathcal{G} such that $L = \mathcal{L}(\mathcal{G})$.

Furthermore, for any regular grammar $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ there exists a (possibly nondeterministic) finite automaton $M(\mathcal{G}) = \langle K, \Sigma, \Delta, s_0, F \rangle$ *naturally isomorphic* to \mathcal{G} , which decides $\mathcal{L}(\mathcal{G})$. The automaton $M(\mathcal{G})$ is defined as follows depending on whether \mathcal{G} respectively is either left-regular or right-regular:

- (i) $K = N \cup \{s_0\}$; (ii) $B \in \Delta(A, a)$ iff $B \rightarrow Aa$ belongs to Π ; (iii) $B \in \Delta(s_0, \alpha)$ iff $\alpha \in \Sigma \cup \{\varepsilon\}$ and $B \rightarrow \alpha$ belongs to Π ; (iv) $F = \{S\}$.
- (i) $K = N \cup F$, with $F = \{s_f\}$; (ii) $B \in \Delta(A, a)$ iff $A \rightarrow aB$ belongs to Π ; (iii) $s_f \in \Delta(A, \alpha)$ iff $\alpha \in \Sigma \cup \{\varepsilon\}$ and $A \rightarrow \alpha$ belongs to Π ; (iv) $s_0 = S$.

$M(\mathcal{G})$ may be deterministic or not depending on structure of \mathcal{G} .

Similarly to how deterministic context-free grammars are defined, a regular grammar \mathcal{G} is *deterministic* if $M(\mathcal{G})$ is deterministic, id est, if \mathcal{G} can be parsed (or implemented) by a deterministic finite state automaton. Deterministic regular grammars are a proper subset of unambiguous regular grammars. As shown in [147], in fact, not every unambiguous regular grammar can be parsed by a (deterministic) finite state machine, even extending our definition of deterministic regular grammars and if a lookahead facility is added to the machine’s capabilities.

Attribute Grammars

Attribute grammars, first introduced by Knuth [160], have been used in syntactic pattern recognition as well as natural language processing and programming languages [7].

Definition 2.6. *An attribute grammar AG is the following four-tuple:*

$$AG = \langle \mathcal{G}, Attr, Func, Pred \rangle$$

where

- $\mathcal{G} = \langle \Sigma, N, S, \Pi \rangle$ is the underlying context-free grammar. N and Σ represent the non-terminal and terminal symbols, respectively, Π is the set of productions, and S is the start symbol.
- $Attr$ is a set of attributes, inherited and synthesized, that are associated with each symbol occurring in the productions in Π . Synthesized attributes are initially given with the terminal symbols, which are passed up the parse tree during parsing, whereas inherited attributes are evaluated top-down from the parents of the node. Each attribute has a type and either represents a specific property of a nonterminal symbol or contains a temporary value exploited in functions or predicates.
- $Func$ is a set of attribute evaluation rules associated with each production $p \in \Pi$. Attributes are evaluated using functions defined on attribute values of symbols in p .
- $Pred$ is a set of semantic conditions associated with each $p \in \Pi$, which are predicates defined on the attribute values.

The semantic conditions impose constraints on the value of the attributes such that the production p can be used only when the conditions are satisfied. This in effect restricts the syntax in addition to the grammar. The language generated by an attribute grammar \mathcal{AG} consists of all strings that have a legal parse tree in \mathcal{AG} (that is, a parse tree in which all attribute values relate in the prescribed way). Because of the predicates, a parse tree of the original context-free grammar may no longer be a legal parse tree of the attribute grammar, and thus the language accepted by an attribute grammar is in general a subset of the corresponding context-free language.

Graph Grammars

Analogously to Chomsky grammars in formal languages theory, graph transformation can be used to generate graph languages. A graph grammar consists of a set of rules, a start graph and a terminal expression fixing the set of terminal graph. This terminal expression is a set $\Delta \subseteq \Sigma$ of terminal labels admitting all graphs that are labeled over Δ .

A *Graph Grammar* is a system $GG = \langle S, P, \Delta \rangle$ where:

- $S \in G_\Sigma$ is the *initial graph* of GG .
- P is a finite set of *graph transformation rules*.
- $\Delta \subseteq \Sigma$ is a set of terminal symbols.

The generated language of GG consists of all graphs $G \in G_\Sigma$ that are labeled over Δ and that are derivable from the initial graph F via successive applications of the rules in P , i.e. $L(GG) = \{G \in G_\Delta \mid S \Rightarrow_P^* G\}$.

2.3.2 Parsing Techniques

In this section efficient parser techniques are described. The concept of parsing schemata will be used because it provide a unifying approach to contextfree parsing [160]. The framework, however, can also be applied to parsing methods that deal with grammars beyond the context-free category. Parsing schemata are inspired by the “item-based” approach to parsing. According to this point of view, recognizing is a process of deducing a final set of items (possibly representing complete parse trees) from an initial set of items (the hypotheses) by means of a set of deduction rules. Different parsing methods use different items (item domains), deduction rules, and sets of initial items. Most parsing methods, including efficient *chart parsers* like Earley and CYK, can effectively be interpreted as item-based parsers.

Let us first define a parsing system, which is defined for a specific grammar and input string.

Definition 2.7. A parsing system P for some grammar G and input-string $a_1 \dots a_n$ is a triple $P' = \langle I, H, D \rangle$, in which

- I is the domain or item set of P , which specifies the allowed items. (The details of the syntax of items may be different per schema.)
- H is a finite set of initial items, or hypotheses. (H needs not be a subset of I .)
- $D \subseteq \wp_{fin}(H \cup I) \times I$ is a set of production steps, where $\wp_{fin}(H \cup I)$ represents all finite elements in the power-set of $(H \cup I)$.

Parsing systems define a parsing method for a specific grammar G and input string $a_1 \dots a_n$. Uninstantiated parsing systems are defined for an arbitrary input string.

Definition 2.8. An uninstantiated parsing system for a grammar G is a function that assigns a parsing system to any $a_1 \dots a_n \in \Sigma^*$. A uninstantiated parsing system is defined by a triple $\langle I, H, D \rangle$, where H is a function that assigns a set of hypotheses to a string $a_1 \dots a_n \in \Sigma^*$.

A function H is usually defined as

$$H(a_1 \dots a_n) = \{[a, i-1, i] \mid a = a_i \wedge 1 \leq i \leq n\}.$$

Definition 2.9. A parsing schema for some (sub)class of contextfree grammars $CG \subseteq CFG$ is a function that assigns an uninstantiated parsing system to any grammar $G \in CG$.

Definition 2.10. For a given $P' = \langle I, H, D \rangle$, the relation $\vdash \subseteq \wp_{fin}(H \cup I) \times I$ is defined by

$$Y \vdash \xi \text{ if } (Y', \xi) \in D \text{ for some } Y' \subseteq Y.$$

Often, $Y \vdash \xi$ is used, instead of $\{\eta_1 \dots \eta_n\} \vdash \xi$, if $Y = \{\eta_1 \dots \eta_n\}$.

Definition 2.11. A deduction sequence for a parsing system $P' = \langle I, H, D \rangle$ is a pair $(Y; \xi_1, \dots, \xi_{i-1}) \in \wp_{fin}(H \cup I) \times I^+$, such that $Y \cup \{\xi_1, \dots, \xi_{i-1}\} \vdash \xi_i$, for $1 \leq i \leq j$.

Definition 2.12. The set of deduction sequences $\Delta \subseteq \wp_{fin}(H \cup I) \times I^+$ for $P' = \langle I, H, D \rangle$ is defined by

$$\Delta = \{(Y; \xi_1, \dots, \xi_j) \in \wp_{fin}(H \cup I) \times I^+ \mid Y \vdash \xi_1 \vdash \dots \vdash \xi_j\}$$

.

Chart Parser

A chart parser is a type of parser suitable for ambiguous grammars, including grammars of natural languages. It uses the dynamic programming approach partial hypothesized results are stored in a structure called a chart and can be re-used. This eliminates backtracking and prevents a combinatorial explosion. Chart parsers can also be used for parsing computer languages. The Earley and *CYK* parsers are a type of chart parser mainly used for parsing in computational linguistics. The parsing schemata will be used to specify such methods.

CYK Parsing

The Cocke Younger Kasami (*CYK*) [61] algorithm determines whether a string can be generated by a given context-free grammar and, if so, how it can be generated. The algorithm employs bottom-up parsing and dynamic programming.

The parsing schema *CYK* is defined by a parsing system P_{CYK} , for all $G \in CNF$ and $a_1, \dots, a_n \in \Sigma^*$. The class of grammars *CNF* is a subclass of *CFG* which is restricted to the so-called Chomsky Normal Form. Since all grammars in *CFG* can be transformed to a grammar in *CNF*, and because of its simplicity, *CYK* is still a much used parsing method. *CYK* can be defined as follows.

$$\begin{aligned} I_{CYK} &= \{[A, i, j] \mid A \in N \wedge 0 \leq i < j\} \\ H_{CYK} &= \{[a, i-1, i] \mid a = a_i \wedge 1 \leq i \leq n\} \\ D^{(1)} &= \{[a, i-1, i] \vdash [A, i-1, i] \mid A \rightarrow a \in P\} \\ D^{(2)} &= \{[B, i, j], [C, j, k] \vdash [A, i, k] \mid A \rightarrow BC \in P\} \\ D_{CYK} &= D^{(1)} \cup D^{(2)} \end{aligned}$$

In the theory of computation, the importance of the *CYK* algorithm stems from the fact that it constructively proves that it is decidable whether a given string belongs to the formal language described by a given context-free grammar, and the fact that it does so quite efficiently. Using Landau symbols, the worst case running time of *CYK* is $\Theta(n^3 \cdot |G|)$, where n is the length of the parsed string and $|G|$ is the size of the *CNF* grammar G .

Earley Parsing

The Earley parser is a type of chart parser mainly used for parsing in computational linguistics, named after its inventor, Jay Earley. The algorithm uses dynamic programming. Earley parsers are appealing because they can parse all context-free languages [160]

The Earley parsing method is inherently left to right. Instead of merely a non-terminal symbol, as with *CYK* items, an Earley item holds an entire production. It uses a dot in the right-hand side of the production to indicate up to which point a parse has completed. The part left of the dot represents the part that is already parsed, and the part right of the dot represents the part that still needs to be done. A dot at the end of a production indicates that the entire production has been recognized. A dot can be moved past a symbol as soon as an item that indicates the completed recognition of that symbol is available. This is captured by the scan and complete deduction rules. The init and predict deductions introduce possible candidates for parsing. The schema is as follows.

$$\begin{aligned}
I_E &= \{[A \rightarrow \alpha \bullet \beta, i, j] \mid A \rightarrow \alpha\beta \in P \wedge 0 \leq i \leq j\} \\
H_E &= \{[a, i - 1, i] \mid a = a_i \wedge 1 \leq i \leq n\} \\
D^{Init} &= \{ \vdash [S \rightarrow \bullet \gamma, 0, 0] \} \\
D^{Predict} &= \{[A \rightarrow \alpha \bullet B\beta, i, j] \vdash [B \rightarrow \bullet \gamma, j, j]\} \\
D^{Scan} &= \{[A \rightarrow \alpha \bullet a\beta, i, j], [a, j, j + 1] \vdash [A \rightarrow \alpha a \bullet \beta, i, j + 1]\} \\
D^{Complete} &= \{[A \rightarrow \alpha \bullet B\beta, i, j], [B \rightarrow \gamma \bullet, j, k] \vdash [A \rightarrow \alpha B \bullet \beta, i, k]\} \\
D_E &= D^{Init} \cup D^{Predict} \cup D^{Scan} \cup D^{Complete}
\end{aligned}$$

This parser yields the following set of recognized items:

$$[A \rightarrow \alpha \bullet, i, j] \mid \alpha \Rightarrow^+ a_1 \dots a_j \wedge S \Rightarrow^+ a_1 \dots a_i A \gamma \text{ for some } \gamma.$$

The parser recognizes the input string if $[S \rightarrow \alpha \bullet, 0, n]$, where $\alpha \Rightarrow^* a_1 \dots a_n$.

The Earley parser executes in cubic time ($O(n^3)$, where n is the length of the parsed string) in the general case, quadratic time ($O(n^2)$) for unambiguous grammars, and linear time for almost all $LR(k)$ grammars. It performs particularly well when the rules are written left-recursively.

Bottom-up Earley

The Earley schema, as presented, is inherently a top-down parser (top-down filtering, see D^{Init} and $D^{Predict}$). This reduces the possibilities for parallelization. We can obtain a purely bottom-up variant of E , by altering the deduction steps of E . The replacement steps for the bottom-up Earley variant, short buE , are given in the following schema.

$$\begin{aligned}
D^{Init} &= \{\vdash [A \rightarrow \bullet \gamma, i, i]\} \\
D^{Scan} &= \{\{[A \rightarrow \alpha \bullet a \beta, i, j], [a, j, j + 1] \vdash [A \rightarrow \alpha a \bullet \beta, i, j + 1]\}\} \\
D^{Complete} &= \{\{[A \rightarrow \alpha \bullet B \beta, i, j], [B \rightarrow \gamma \bullet, j, k] \vdash [A \rightarrow \alpha B \bullet \beta, i, k]\}\} \\
D_{buE} &= D^{Init} \cup D^{Scan} \cup D^{Complete}
\end{aligned}$$

This parser yields the following set of recognized items:

$$\{[A \rightarrow \alpha \bullet, i, j] \mid \alpha \Rightarrow^+ a_i \dots a_j\}.$$

The set of correct final items is identical to the set associated with Earley (E). Obviously, since the D^{Init} of buE produces more items than the D^{Init} and $D^{Predict}$ of E do together, the set of items buE recognizes is larger than with E . This variant, however, is better suited for parallel processing.

Recursive Transition Networks

A *Recursive Transition Network* (RTN) is a graph theoretic structure used to represent the rules of a context free grammar. A RTN is a directed graph with labeled states and arcs, a distinguished state called the start state, and a distinguished set of states called final states. It looks essentially like a non-deterministic finite state transition diagram except that the labels on the arcs may be state names as well as terminal symbols [191]. RTNs have application to programming languages, natural language and lexical analysis. Any sentence that is constructed according to the rules of an RTN is said to be “well-formed”.

An *Augmented Transition Network* (ATN) is a *Recursive Transition Network* (RTN) [191] enriched by functions and predicates that respectively compute and constraint attribute values.

ATNs are used especially in parsing relatively complex natural languages, and having wide application in artificial intelligence. An ATN can, theoretically, analyze the structure of any sentence, however complicated.

Recursive Transition Networks build on the idea of using finite state machines (Markov model) to parse sentences. W. A. Woods in [191, 50] claims that by adding a recursive mechanism to a finite state model, parsing can be achieved much more efficiently. Instead of building an automaton for a particular sentence, a collection of transition graphs are built. A grammatically correct sentence is parsed by reaching a final state in any state graph. Transitions between these graphs are simply subroutine calls from one state to any initial state on any graph in the network. A sentence is determined to be grammatically correct if a final state is reached by the last word in the sentence.

This model meets many of the goals set forth by the nature of language in that it captures the regularities of the language. That is, if there is a process that operates in a number of environments, the grammar should encapsulate the process in a single structure. Such encapsulation not only simplifies the grammar, but has the added bonus of efficiency of operation. Another

advantage of such a model is the ability to postpone decisions. Many grammars use guessing when an ambiguity comes up. This means that not enough is yet known about the sentence. By the use of recursion, ATNs solve this inefficiency by postponing decisions until more is known about a sentence.

2.4 Knowledge Representation

2.4.1 Ontologies

Ontologies [172] enable to directly encode domain knowledge in software applications, so ontology-based systems can exploit the meaning of information for providing advanced and intelligent functionalities. Since ontologies are widely used to represent knowledge or meaning they are often seen as providing the backbone for the Semantic Web [83, 22]. Moreover, one of the most interesting and promising application of ontologies is information extraction from unstructured documents.

Definition 2.13. *let Z be a set of constants and \tilde{Z} (the whole set of values) be that one obtained by the union of Z with all finite lists of elements in Z . An Ontology on Z is a 9-tuple:*

$$\mathcal{O}_Z = \langle D, A, C, R, I, \leq, \sigma, \varphi, \iota \rangle$$

where:

- D, A, C, R are finite and disjoint sets of entity names respectively called data-types, attribute-names, classes and relations. Set D contains only integer and string data-types. Set A contains the special attribute-name **id**. Elements in $C \cup D$ are called flat-types. For each flat-type t there is a list-type denoted by $[t]$. The union of all flat-types and list-types is called the set of types and denoted by T .
- $I = 2^{A \times \tilde{Z}}$ is the set of instances in \mathcal{O}_Z . $I = I_c \cup I_r$, where I_c are called objects, whereas I_r are called tuples.
- \leq is a partial order (called isA) on C .
- $\sigma : C \cup R \rightarrow 2^{A \times T}$ is the schema function. Let e be an element in $C \cup R$. The set $\sigma(e)$ is called the schema of e and any couple $\langle a, t \rangle \in \sigma(e)$ is the (schema) attribute of e with name a and type t . The schema of a class c contains the attribute $\langle \mathbf{id}, c \rangle$, whereas no relation schema contains attributes with name **id**. Any schema contains only attributes with distinct names.
- $\varphi : D \rightarrow 2^Z$ is the domain function that associates to each data-type $t \in D$ a domain value in Z .
- $\iota : C \cup R \rightarrow 2^I$ is the instance function that associates to each class or relation its subsets of instances. When ι is applied to C the elements in the set $\iota(C) = I_c$ are called objects, whereas the elements in the set $\iota(R) = I_r$

are called tuples. Let e be an element in $C \cup R$ and $\hat{i} \in \iota(e)$ be an instance of e , any couple $\langle a, z \rangle \in \hat{i}$ is the (instance) attribute of \hat{i} with name a and value z . Given an instance $\hat{i} \in \iota(c)$ of a class c , the value z of the attribute $\langle \mathbf{id}, z \rangle \in \hat{i}$, is called the object identifier (oid) of \hat{i} .

In the following ontology constructs are explained by means of a running example and exploiting the syntax of the DLP+ ontology representation language [158].

Ontology schemas

A *class* is a name and an ordered list of attributes (attribute **id** is implicitly declared) identifying the properties of its instances. Thus, a class can be thought as an aggregation of individuals (objects) that have the same set of properties (attributes). Each attribute is identified by a name and has a type specified as a data-type or user-defined class. Class schemas expressing knowledge related to weather can be declared as follows:

```
class koppenClimate (lTempF:integer,hTempF:integer,avgRainfallCm:integer).
  class continentalClimate (summerHumidity:string,winterHumidity:string)
    isa {koppenClimate}.
class place (name:string).
  class state (areaKm2:integer,capital:city,neighborState:[state])
    isa {place}.
  class city (population:integer,inState:state) isa {place}.
```

Classes *place* and *city* show how class hierarchies (taxonomies) can be built up by using **isa** key-word. Class *city* shows the ability to specify user-defined classes as attribute types (i.e. **inState:state**). Class *state* has a list-type attribute (i.e. **neighborState:[state]**).

Relationships among objects are represented by means of *relations*, which like classes, are defined by a name and a list of attributes. Relation *cityClimate*, which asserts what is the climate of a given city, can be declared as follows:

```
relation cityClimate (c:city, climate:koppenClimate).
```

Ontology Instances

Objects are declared by asserting new facts, they are unambiguously identified by their *object-identifier* (*oid*) and belong to a class. By considering the weather example, instances for the classes *continentalClimate*, *state* and *city* can be declared as follows:

```
hotSummer:continentalClimate (lTempF:-36,hTempF:86,avgRainfallCm:90,
  summerHumidity:"dry",winterHumidity:"wet").
illinois:state (name:"Illinois",areaKm2:140998,capital:Springfield,
  neighborState:[wisconsin,kentucky,iowa,missouri,indiana]).
chicago:city (name:"Chicago",population:2833321, inState:illinois).
sterling:city (name:"Sterling",population:15451, inState:illinois).
```

For class `state` is represented an object which `oid` is the constant `illinois`, while string `"Illinois"` is the value for the attribute `name:string`. The attribute `neighborState` contain a list of `oids` that represent Illinois neighbors. Tuples are defined by a name and a list of attribute values (attribute names are optional). In the weather example the tuple of the relations `cityClimate` which asserts that the climate of the city `chicago`, is `hotSummer` can be declared as follows:

```
cityClimate(c:chicago, climate:hotSummer).
```

Inference Rules

It is worth noting that DLP+ [109, 158] is actually an extension of Disjunctive Logic Programming [58] (DLP, which has been enriched by concepts from the object-oriented paradigm. So, it makes possible to specify complex inference rules and constraints over knowledge base schemas and instances, merging, in a simple and natural way, the declarative style of logic programming with the navigational style of ontologies. In particular, the approach allows to infer tuples and objects and define global integrity constraints called *axioms*.

Reasoning Modules. Given an ontology, it can be very useful to reason about the data it describes. Reasoning modules are the language components endowing DLP+ with powerful reasoning capabilities to DLP+. Basically, a reasoning module is a disjunctive logic program conceived to reason about the data described in an ontology. Reasoning modules in DLP+ are identified by a name and are dened by a set of (possibly disjunctive) logic rules and integrity constraints. For instance,

```
module(hotCitiesIllinois){
  hotcity(X):-cityClimate(c:X, climate:hotSummer),
             X: city(instate:illinois).
}
```

Queries. An important feature of the language is the possibility of asking queries in order to extract knowledge contained in the ontology, but not directly expressed. As in DLP a query can be expressed by a conjunction of atoms, which, in DLP+, can also contain complex terms. For instance, we can ask the cities in illinois state as follows:

```
X: city(instate:illinois)?
```

Consistency and Semantics

Consistency of schemas. Any schema contains only attributes with distinct names. Let c_1 and c_2 be two classes such that $c_2 \leq c_1$. For each attribute $\langle a, t_1 \rangle \in \sigma(c_1)$ there exists precisely one other attribute $\langle a, t_2 \rangle \in \sigma(c_2)$ with the same name. If t_1 is either a data-type or a list-type $[t]$, where t is a data-type, then $t_2 = t_1$. If t_1 is a class, then t_2 is a class and $t_2 \leq t_1$ holds. If t_1 is a

list-type $[\hat{t}_1]$, where \hat{t}_1 is a class, then $t_2 = [\hat{t}_2]$ and $\hat{t}_2 \leq \hat{t}_1$ holds.

Consistency of instances. Given a type $t \in T$, then value z is *compatible* with t if one of the following conditions holds: (i) $t \in D$ and $z \in \delta(t)$; (ii) $t \in C$ and z is an *oid* of an instance of t ; (iii) $t = [\hat{t}]$ is a list-type, and z is a list of values all of which are *compatible* with \hat{t} . Set \tilde{Z}^t denotes all values in \tilde{Z} compatible with t . Let e be an element in $C \cup R$ that has schema $\sigma(e) = \{\langle a_1, t_1 \rangle, \dots, \langle a_h, t_h \rangle\}$. Then, each instance $\iota(e)$ of e has the following form $\{\langle a_1, z_1 \rangle, \dots, \langle a_h, z_h \rangle\}$ where each z_j is *compatible* with t_j ($1 \leq j \leq h$). There are no two instances sharing the same *oid*.

Semantics The semantics of \mathcal{O}_z is given in terms of Datalog. The datalog representation $\mathcal{C}(\mathcal{O}_z)$ of \mathcal{O}_z is the set of all the clauses created from \mathcal{O}_z as follows: (i) for each element $e \in C \cup R$ and instance $\hat{t} = \{\langle a_1, z_1 \rangle, \dots, \langle a_h, z_h \rangle\}$ in $\iota(e)$, create ground fact $e(z_1, \dots, z_h) \leftarrow$; (ii) for each couple of classes c_1, c_2 in C such that $c_2 \leq c_1$, create clause $c_1(X_1, \dots, X_h) \leftarrow c_2(X_1, \dots, X_h, \dots, X_k)$ where $k \geq h$, $|\sigma(c_1)| = h$, and $|\sigma(c_2)| = k$.

Given a Datalog query \mathcal{Q} where any of its predicate symbol $e \in C \cup R$ has arity $|\sigma(e)|$. So, the set of couples $\langle p(X_1, \dots, X_n), \mathcal{Q} \rangle$, where p is a generic predicate, is called *decision query* on \mathcal{O}_z . Whenever each variable X_i assume value z_i ($1 \leq i \leq n$), thus expression $\mathcal{C}(\mathcal{O}_z) \cup \mathcal{Q} \models p(z_1, \dots, z_n)$ may be evaluated³ for checking whether the ground atom $p(z_1, \dots, z_n)$ is true with respect to \mathcal{O}_z .

³ Lists of elements are handled as constants.

Part II

Information Extraction: State Of Art

*Data is a precious thing and
will last longer than
the systems themselves.
~ Tim Berners-Lee ~*

Information Extraction (IE) is the process aimed at converting semi and unstructured document into a specific structured format in order to make information available for subsequent manipulation or integration steps. *Classical IE* include named-entity recognition (identifying persons, places, organizations, etc.) and relationship extraction between named entities. *Web information extraction* is the application of IE techniques to process the vast amounts of unstructured content on the Web. Information extraction from Web sites is often performed using wrappers. A *wrapper* is a program that extracts data from information sources of changing content and transform unstructured input into structured output formats, normally XML. A *wrapper generation systems* describes the transformation rules involved in such transformations. There is wide agreement that wrapping should exclude operations related to data transformation and integration tasks that may follow extraction.

In the last years much research has been done for supporting information extraction, specially on the web, and several approaches have been proposed. There exist many surveys about web data extraction tools (e.g. [18, 64, 37, 107]), flat text and ontology based information extraction tools (e.g. [189]), and some specific surveys about the table recognition problem (e.g.[196]).

There are a number of ways to classify wrapping systems. A complete categorization was made by Laender et al. [107]. They proposed the following taxonomy:

- Wrapper induction tools (e.g., DEPTA [197, 198], IEPAD [38], WIEN [133], LIXTO Visual Wrapper [17, 75, 19], SoftMealy [92], STALKER [129]) that are based on the structural information and formatting features of the web pages, and generate extraction rules derived from a given set of training examples or pattern discovery techniques.
- HTML-aware tools (e.g., LIXTO [17, 75, 19], W4F [164], XWrap, Road-Runner [42]) that rely on the structural information of the web pages, too, and use HTML parse trees for creating extraction rules.
- Modeling-based tools (e.g., DEByE [106], Robosuite, NoDoSE) that locate in the web pages portions of data conforming to a predened structure provided according to a set of modeling primitive as tuples or lists.
- NLP-based tools (e.g., WHISK, RAPIER, SRV) that work on phrases and sentences elements within the web pages to derive extraction rules, by applying techniques such as ltering, part-of-speech tagging and lexical semantic tagging.
- Ontology-based tools (e.g., BYU), that rely on the data and not on the structure of the source documents. These tools use ontology to locate constants in the page and to construct objects with them.

Many of these approaches such as Lixto, Wargo and Fetch (based on Stalker), which started out in the academic domain, are now fully edged commercial products.

In this dissertation information extraction approaches will be mainly classified considering the underlying structure which they are able to work on and if they are able to consider the the semantics of information:

- Classical Information Extraction Systems, that works on the text.
- Web Information Extraction Systems, that works on HTML documents and generally make use of the tree based internal structure.
- PDF Information Extraction Systems, that are few and recent works aimed at analyzing, recognizing and extracting information from the PDF documents.
- Ontology-based Information Extraction Systems, that exploits the knowledge for recognizing information contained in flat text, in order to populate an output ontology.
- Query Languages, that are used to extract data using logic and/or spatial information.

Moreover, as described in [37], the automation degree adopted in the denition of wrappers and extraction rules is considered. So, existing approaches and systems, can be classied in manual, semi-supervised and unsupervised (fully automatic):

- Manual techniques require the use of a wrapper programming language such as PiLLOW [30] or Jedi [93] and have fallen out of popularity in recent years, as construction and maintenance of wrappers is time-consuming and their specification language presents a significant learning curve. Most significant *manual* approaches are: TSIMMIS, Minerva, Web-OQL , W4F, XWRAP [107, 37], JEDI [93], FLORID [120].
- Semi-automatic systems allow the generation of wrappers based on user input. These approaches can broadly be split up into systems based on wrapper induction and systems based on wrapper specification. There is currently a wide range of wrapper induction systems in the literature, such as WIEN [133], and Stalker [129], which use machine learning techniques to generate a wrapper program from a set of training examples provided by the user. In systems based on wrapper specification, the user is directly involved in programmatically creating the wrapper. In contrast to manual systems, wrappers are usually generated based on a few examples in an intuitive and interactive way. Examples of such systems include Lixto [17], DEByE [106] and Wargo [156]. Semi-automatic approaches have also been referred to as supervised approaches in the literature. In order to clarify the distinction between approaches using supervised learning or wrapper induction and approaches based on wrapper specification, we refer to the latter as user-guided wrapper generation. These approaches could be seen as combining the power and precision of manual approaches with the user-friendliness and shallow learning curve of automatic systems.
- Fully automatic systems generate wrappers without any user input. The RoadRunner approach [42] employs a matching technique to generate a

wrapper based on several examples of documents from the same class and therefore does not require any labelled examples. ExAlg [10] works in a similar fashion. The MDR (Mining Data Records) approach uses string-matching [114] and partial tree alignment [197] techniques to locate examples of repeating data records on a Web page. The work of [59] uses an ontological model for a specific domain to extract data from arbitrary Web pages about this domain. In general, the advantage of fully automatic approaches is that they can be applied to any arbitrary document that has not been encountered before. However, they are neither as precise, nor can they extract such detailed information as semi-automatic approaches.

Finally, an important considered characteristic is the exploitation of spatial relations to locate the desired data instances (e.g. [104], the graph-based approach to wrapping PDF documents [85]).

Flat Text Information Extraction

Classical problems in information extraction include named-entity recognition and relationship extraction among entities. Historically, information extraction [80] was studied by the Natural Language Processing community in the context of identifying organizations, locations, and person names in news articles and military reports. The first systems uses regular expressions, simple grammars and finite-state automata. The first very evaluation of IE task was made from 1987 to 1998 during the Message Understanding Conferences (MUC) [80] sponsored by the U.S. Defense Advanced Research Projects Agency (DARPA). During the MUC (6 and 7) four specific evaluations were performed: Named entity, coreference, template element, template relation. Now the the MUC tasks are part of Automatic Content Extraction (ACE) program ¹ of the U.S. National Institute of Standards and Technology (NIST). The ACE includes the following tasks:

- Entity Detection and Recognition. An entity is an object or a set of objects in one of the semantic categories of interest (Like persons, organizations, locations, etc.).
- Relation Detection and Recognition. Such a task is aimed at found semantic relations between pairs of entities. ACE defined some type of relations.
- Event Detection and Recognition. Such a task is aimed at actions. ACE defined some type of events (Like about movement, conflicts e.g. “The bandit killed a man”.)
- Entity Translation between different languages.

In particular, entity extraction refers to the identification of mentions of entities (such as persons, locations, organizations, phone numbers, etc.) in unstructured text. The annotation involve a large number of rules and dictionaries. For example, a same person can be identified by shorthands, and formats used in person names. Binary relationship extraction refers to the task

¹ The ACE task description is available at <http://www.nist.gov/speech/tests/ace/> and the ACE guidelines at <http://www ldc.upenn.edu/Projects/ACE/>

of associating pairs of named entities by a relation. For instance for extracting instances of the `CompanyCEO` relationship, intent to found the pair person and company such that the person is CEO of a company. Assuming that persons and organizations have already been annotated, the following rules allow for identifying `CompanyCEO` relationship instances.

```
<CompanyCEO> <- <Person> RegExp('\s+CEO\s+ of \s,') <Organization>
<CompanyCEO> <- <Organization> RegExp('\s+CEO\s+') <Person>
```

In the regular expressions `\s+` represent an arbitrary amount of whitespace separating the individual words of the phrase.

In the following Sections, we distinguish between two common information extraction techniques from text based on: pattern matching and machine learning.

3.1 Pattern Matching Based IE

Many IE systems during MUC evaluation use high-accuracy rules, dictionaries and patterns for each specific domain. The rule-based extraction programs are often called annotators because the output is not only the extraction of the information, but the concepts of *span* is also used. A span corresponds to a substring of the document text represented as a pair of offsets (begin, end). A rule is of the form $A \leftarrow P$, where A is an annotation and P is a pattern specification. Evaluating the rule associates any span of text that matches pattern P with the annotation A .

Manually writing and editing patterns requires some skill and considerable time. So some systems have moved on to learning these patterns automatically based on an annotated corpus pre-processed by syntactic and semantic analyzers. A more comprehensive survey of pattern matching based IE approaches can be found in [128]. However, for particular domain a separate annotated corpus is needed. Therefore some systems have used unsupervised learning approach [159, 193, 173]. Natural language techniques are often exploited.

3.2 Machine Learning Based IE

Pattern matching approach is not enough for obtaining resilient systems that can be used into new domains. Therefore, IE approach is split in a pipeline of sub-tasks (e.g. name identification and classification, parsing, relation extraction) and for each subtask a machine learning method is applied [32, 70, 171, 67].

IE technology has benefited from improvements that have been brought mainly from Machine Learning (ML) techniques [177, 41, 46]. The statistical learning paradigm, which employs generative models (e.g., Hidden Markov

models - HMMs) and discriminative models (e.g., Conditional Random Fields - CRFs), has recently attracted increasing interest for IE tasks. The General Architecture for Text Engineering (GATE) [43] is a widely used NLP framework and provides an easy-to-use platform to employ this technique.

Web Information Extraction

Initial Wrapper allowed information extraction from free text. Textual documents are unstructured documents and requires natural language processing techniques such as Part-of-Speech tagging. Nowadays, with the advent of the Web, Wrapper generation systems focus on the semi-structured format. HTML tags allow to define relations between block of information. However, it is not enough considering only the internal format. In fact, there aren't strict rules on the semantics of tags and can be used for represent structural or layout information. In this dissertation, the Web IE tools are classified considering the automation degree.

4.1 Manual IE

First IE approaches in the Web domain have seen systems where wrappers were constructed manually. For instance, TSIMMIS [82] proposes a framework for the manual construction of Web wrappers. In TSIMMIS a wrapper takes as input a specification made by a sequence of commands given by programmers describing the pages and how the data should be transformed into objects. Commands take the form (variables, source, pattern), where source specifies the input text to be considered, pattern specifies how to find the text of interest within the source, and variables are a list of variables that hold the extracted results. The generated outputs are represented using the Object Exchange Model. The output is composed by the target data and by additional information about the result.

4.2 Semiautomatic IE

Manual wrapper systems did not scale well and were inexible to change in document structure. A lot of IE systems have been developed to semi-automatically generate wrappers based on two main approaches: (i) wrapper induction (ii) visual wrapper specification.

4.2.1 Wrapper Induction Systems

Wrapper induction systems resort to ML techniques for generating extraction rules. Wrapper induction allow for learning wrapper programs from examples. Kushmerick [105] defined the wrapper induction problem as a simple model of information extraction. Extraction rules are learned from these examples, thus extraction accuracy relies on the number and the “quality” of the examples. Wrapper induction systems do not require the human to manually write extraction rules, but to label a number of documents and have machines inductively learn the extraction rules. There exist many wrapper induction systems, new approaches appear in literature [113] and several wrapper induction systems are successfully commercialized.

There not exists standard datasets such that different Wrapper Induction System effectively can be compared. The most known data set used in information extraction is RISE ¹ However it is too old (before 2000) and can be considered out-dated with respect to the current structure and layout of modern Web pages.

The work [94] generate node predicates on the DOM tree representation of a HTML document (such as tagName, tagAttribute, #children, #siblings). These are constructed for each example node in the training set together with those nodes along the path towards the root from each such node - until a common ancestor is reached. Verification rules are generated from special verification predicates to be checked later. Finally, patterns with equal extraction results are combined.

In [33], the authors also use the DOM tree representation of web documents to detect repetitive occurrences of pre/post subtree constellations. They use boolean logic to combine all rules into a minimal disjunctive normal form over all training examples. Applying supervised learning to Web wrapper generation bears a crucial drawback: sufficiently large sets of documents with annotated examples are hard to obtain and require expertise in the making. Labor and time investments renders the labeling of examples costly, creating a bottleneck in IE from Web documents. As a result, part of the research focused on tools that allow non-expert users to create wrappers interactively via visual user interaction. Starting with a set of non-annotated documents, the algorithm tries to infer the correct wrapper by asking the user to annotate a minimal number of examples.

The STALKER system [129] introduced hierarchical wrappers generated from landmark automata. Stalker generates so-called landmark automata (SkipTo functions) from a set of training data. Nesting patterns allows the system to decompose complex extraction rules into a hierarchy of more simple extraction rules. Stalker employs a covering algorithm to reduce its initial set of candidate rules to a best-fit set of rules. Co-testing learns both forward

¹ <http://www.isi.edu/info-agents/RISE>. RISE is a collection of documents (HTML and flat text) related to seminar announcements, Okra search, IAF address finder.

and backward rules on the document. In case of agreement the system knows that the selected rule is correct, otherwise the user is given the conflicting example for labeling in order to resolve the conflict (active learning).

DEByE [106] is similar to STALKER as it also used landmark automata for rule generation. It applies bottom-up extraction rules. Unlike other wrapper induction systems that typically work top-down, their approach allows for more flexibility as the order of individual patterns here is no longer relevant.

SoftMealy system [92] Softmealy is based upon a finite-state transducer and it utilizes contextual rules as extraction rules. Before being input to the transducer, a document is tokenized and special separators are added between two consecutive tokens. The transducer regards its input as a string of separators, where state transitions are governed by the left and right context tokens surrounding the current input separator. SoftMealy machines have two learnable components: the graph structure of the state transducer and the contextual rules governing the state transitions. The graph structure should be kept simple and small, which is achieved via a conservative extension policy. Contextual rules are learned by a set-covering algorithm that finds the minimal number of contextual rules covering all examples.

In boosted wrapper induction BWI [71] delimiter-based extraction rules are generated from simple contextual patterns. BWI works both in natural language and wrapper domains. It uses a probabilistic model for tuple length classification together with a boosting algorithm for delimiter learning.

4.2.2 GUI-based Web Data Extraction

Visual wrapper generation is based on interactive tools to assist the wrapper designing. These tools are usually equipped with graphical interfaces by which the user visually specifies what information the system has to extract. Thus, the goal of Graphical User Interfaces for Visual Web data extraction is not only to show results in a presentation format, but allow the selection of interesting information, that the user want to extract removing all presentational features, so that only pure concepts remain. Web data extraction exploiting GUIs are variants of wrapper induction systems. Like classical inductive systems GUI-based extraction systems do not require the human to manually write extraction rule. While classical inductive techniques use machine learning techniques for the generalization and conjunction of extraction patterns, GUI-based approaches allow user to specify what he intends to extract in a comfortable way considering the WYSIWYG paradigm. Therefore such approaches are aim at user-friendly and non-expert users. All systems have a similar GUI that: show the HTML browser; allow for selecting the interesting data by mouse; allow for navigating the DOM tree of HTML pages; exploit internal extraction rules based on path expressions. The extraction process is addressed to template of documents, such as page describing products, news articles, data of people. For each template a set of extraction rules is generated and information can be extracted and stored in structured format. The

user select information (for instance names and prices of products) that have to extract and assign labels them. The extraction rules, which represent a wrapper, are inferred by the selected information.

WIEN [133] is the first system for wrapper induction from semi-structured documents. The system learns extraction rules in two stages: In a first step, simple LR rules are generated from the training data, that specify left and right delimiters for each example. The second step refines the LR rules into example conforming HLRT rules, where additional Head and Tail delimiters are added if necessary. The WIEN algorithm assumes attributes to be equally ordered among all tuples, and it cannot deal with missing attributes. WIEN supports both single-slot and multi-slot patterns.

Sahuguet et al presented W4F system [164] that uses a SQL-like language named HEL. In W4F parts of the query in HEL can be generated using a visual wizard which returns the full XPath location path of DOM nodes. So the HEL language is unable to recognize when information in a Web page is presented by the same visual pattern but is represented by different tag structures. So, the W4F system may benefit from using SXPath as more expressive basis for defining extraction rules.

XWRAP [115] is a wrapper generation framework. XWRAP uses a common library to provide basic building blocks for wrapper programs. In this way, tasks of building wrappers specific to a Web source are separated from repetitive tasks for multiple sources.

NODOSE [3] extracts information from plain string sources and provides a user interface for instance labeling. NoDoSe is an interactive tool that allows the user to hierarchically decompose a complex wrapper pattern into simple sub-wrappers. The system works both in the free text and the Web domain with the help of two distinct mining components. Learning rules consist of start and stop delimiters, either on text level (first component) or on HTML structure level (second component). Parsing generates a concept tree of rules that support multi-slot patterns and attribute permutation.

LIXTO [17, 75, 19] was started by Gottlob et al. at TUWIEN and is now developed and sold by the LiXto GmbH software house. is a method for visually extracting HTML/XML wrappers under the supervision of a human designer. LiXto allows a wrapper to interactively and visually define information extraction patterns on the base of visualized sample Web pages. These extraction patterns are collected into a hierarchical knowledge base that constitutes a declarative wrapper program. The extraction knowledge is internally represented in a Datalog-like programming language called Elog [16]. The typical user is not concerned with Elog as wrappers are build using visual and interactive primitives. Wrapper programs can be run over input Web documents by a module in charge of extraction which then translates the output in XML.

DENODO formerly known as Wargo, is a platform for creating and executing navigation and extraction scripts that are loosely tied together. It offers graphical wizards for configuring wrappers and allows DOM events to be pro-

cessed while navigating web pages. Deep Web navigation can be executed in Internet Explorer, and the result pages are passed on to the extraction program. Furthermore, Denodo offers some wrapper maintenance functionalities. It additionally offers a tool called Aracne for document crawling and indexing.

4.3 Automatic IE

Actual Wrapper Induction System do not scale when is required the extraction of large amounts of information from different Web site, in particular when have different html code. In fact, are needed a large number of examples. Therefore, unsupervised extraction is necessary in this case. Fully-automatic Web data extraction aim at extracting relevant information from HTML documents, without requiring human intervention throughout the process.

Automatic IE approaches can be classified in according to the goal that they try to resolve in:

- *cleaning approach* [90] that allow for extracting a block of flat text containing the required information (e.g. extract news), Such approaches generate wrapper that returns the flat text, without assigning labels. There was a competition named CLEANVAL² and part of the “Web as Corpus” initiative (WAC) on May 2007. The objective of WAC is to collect massive textual information from the Web in order to use it for natural language processing (NLP) and linguistic research, forming representative background corpora and language models. On the competition site a dataset is available and it represent a benchmark for cleaning approaches.
- *records extraction* that allow for extracting repetitive records (e.g. list of products and search engine results) [91]. They exploit the recurring patterns that describe the set of records and allow for recognizing different items of each record. They can consider or the flat text or the internal document format of HTML pages, or the visual cues.

For automatic extraction of search engine results, several smaller datasets are available, among those the Omini dataset (Available from Sourceforge via <http://sourceforge.net/projects/omini/>) and MDR collection. None of them may count as standard dataset, though.

Some relevant automatic information extraction systems are described in the following.

ROADRUNNER [42] was developed at the University of Roma³ and applies to intensive Web sites, i.e. those sites with large amounts of data and a rather regular structure. RoadRunner works by comparing the HTML structure of a set of sample pages of the same type, and generates a schema for the data contained in the pages. This schema is used as a starting point for the inference of a grammar which is capable to recognize the instances of attributes

² <http://cleanval.sigwac.org.uk/>

identified for this schema in the set of sample pages. The extraction procedure is based on an algorithm that compares the tag structure of the set of sample pages and produces regular expressions able to handle structural differences found in the set of sample pages. A peculiar feature of RoadRunner is that this procedure is completely automatic and no user intervention is required.

OMNI [28] parses Web pages into tree structures and performs object extraction in two stages. First, it uses a set of subtree extraction algorithms to locate the smallest subtree that contains all the objects of potential interest. Second, it employs a suite of object extraction algorithms to find the correct object separator tags that can effectively separate objects. The prototype supports five separator tag identification heuristics, covering a wide range of possible mechanisms for discovering object separators. Each one of the five heuristics independently ranks the candidate tags. Both stages are fully automated and require no human assistance.

MDR [114] identifies data regions by searching for multiple generalize-nodes using edit-distance similarity where generalize-nodes are a fix combination of multiple child nodes and their corresponding subtrees. MDR does not identify the most relevant data records but rather reports each repetitive sub-region contained in a Web page. Recently the authors proposed an improvement of their system named DEPTA (MDR-2) [197, 198] operating on a tag tree built according to visual rendering information. Additionally they mention that gap information is incorporated to eliminate false node combinations but nothing is said about the realization. Finally they proposed an approach for data record alignment by progressively growing a seed tag tree. The alignment is partial because only these nodes of a data record become aligned whose position for inserting into the seed tree can be uniquely determined.

STAVIES [150] employs clustering techniques to segment the Web pages and locate the region that contains the data records as well as the boundaries separating them. This method is restricted by using the cardinality of common ancestors of two nodes as the similarity measure.

4.3.1 Visual-based Approaches

Some newer Wrapper generators also exploit visual cues by analyzing HTMLs rendered by a Web browser engine [178].

VINTS [201] automatically generates search result record extraction rules using visual context features and tag structure information. To this end, ViNTS first analyzes the graphical representation without considering the tag structure to identify content regularities by means of so-called content lines. Next, structural regularities among HTML blocks are combined with these visual features to generate wrappers. To weight the relevance of different extraction rules, visual and non-visual features are considered.

ViPER [169] system builds on similar techniques as ViNTS, but extends its capabilities by not only allowing to identify recurring blocks that are aligned

vertically, but also those aligned in a horizontal fashion. Next to the exploitation of visual cues, ViPER used multiple sequence alignment techniques, known from bio-informatics, to identify structure and patterns in HTML tag sequences.

VIDE [116] system adopts a strategy for representing deep web pages very close to our data model but it do not exploit also the HTML structure. We must learn the wrapper by using the spatial data model and clustering algorithm. With just one processing we should have both data region, data records and data items. A deep comparison with this system is required. Take also into account experimental result obtained by this system.

This last three system exploit heuristic algorithms based on visual information and do not consider internal structure. However, we believe that internal structure could, if correctly processed, be used to improve the robustness of automatic information extraction systems.

PDF Information Extraction

There is a substantial interest from industry in wrapping documents in formats such as PDF and PostScript. The intrinsic print/visual oriented nature of PDF encoding poses many issues in defining “ad hoc” IE approaches. In fact, wrapping of such documents must be mainly guided by a visual reasoning process over white space, which is substantially different from web wrapping. While many HTML-oriented approaches and systems are available, just few approaches that deal with PDF document formats have been proposed in literature. In this Section, firstly document understanding problem will be shortly described, then table recognition and wrapping problem from PDF documents will be discussed.

5.1 Document Understanding

In PDF files the documents logical structure is not explicitly present. So, it could be useful to make use of document analysis approaches in order to rediscover this structure and allow for locating the data to be wrapped. Document image analysis and understanding can be grouped considering their approaches in: bottom-up and top-down.

- Bottom-up Pixel-based Approach. Document image analysis has its origins in the OCR community, and there is a huge amount of literature describing document analysis systems that work on scanned images of pages [5, 9, 165, 131, 131] The vast majority of such systems use a bottom-up segmentation process: the image is first deskewed and binarized (or thresholded), and segmentation then is performed using pixel-based operations.
- Top-down based Approach. Earlier segmentation algorithms include top-down methods [130, 81] In this method, the page is recursively “cut” in half across a visually salient boundary, usually whitespace or a ruling line. The main weakness of this approach is that it is not able to segment all page layouts completely. Specially, the page layout must be X-Y decomposable.

ASCII format (and similar xed-width text formats) allows only a very restricted range of layout conventions. The work [161] uses a bottom-up method to merge lines of text, as obtained from an OCR package, into a hierarchical structure based on repeating indentation patterns. This technique assumes no prior knowledge about the formatting conventions being used. Multi-column and more complex layouts are not addressed, and are a rarity in this format. This paper also describes techniques in table detection and graphics recognition.

5.2 PDF Table Recognition

Table recognition and extraction (TRES) has recently received as considerable attention as it can be considered a “per se” research field. Therefore, a large body of work concerning approaches and systems aimed at recognizing and extracting tables from documents having different internal encodings is currently available in literature. A survey due to Zanibbi et al. [196], provides a detailed description of already existing TRES approaches and systems. TRES approaches can be classified by using different criteria in: (i) *predefined layout-based, heuristics-based, and statistical-based*, by considering the adopted recognition and extraction method, as described in Wang [186]. These approaches use different knowledge engineering techniques founded on machine learning and statistics. (ii) *table recognition-oriented, labeling-oriented and cell classification-oriented*, by considering which kind of problem approaches aim at solving, as described in Pivk et al. [153]. (iii) *HTML-oriented, PDF-oriented and flat text-oriented*

While many HTML-oriented TRES approaches and systems are available (e.g. [72, 110, 104]), just few approaches that deal with PDF and flat text document formats have been proposed in literature (e.g. [11, 88, 101, 194]). Furthermore approaches working on PDF and flat text documents are less performing than those working on HTML documents. This is due to the highly unstructured nature of PDF and flat text documents that pose complex issues in defining TRES methods. Currently, there is an increasing interest toward print-oriented documents.

Yildiz et al. [194] describe a system called *pdf2table* based on an heuristic approach that performs two main tasks: table recognition, in which information organized in tabular structure are recognized, and table decomposition in which recognized elements are assigned to a table model. The table recognition task works on the output of *pdftohtml* tool and tries to assign text blocks extracted by *pdftohtml* in containers called multi-lines and multi-lines block. At the end of the process a multi-line block could contain a table. The table decomposition task tries to split multi-line blocks in column in order to define the table structure. Authors provide an experimental evaluation of the approach based on a dataset composed by 150 documents, but the dataset is

not available and no rigorous definitions of lucid and complex table is given. so, a direct comparison with such an approach is impossible.

Hassan et al. [88] propose a method for detecting tables in PDF files. They group tables into three categories: tables with both horizontal and vertical ruling lines, tables which contain only horizontal ruling lines, and tables where ruling lines are absent. The first group appears to be the most reliable in detecting and understanding tables because it exploits ruling lines. By using a principle of rectangular containment, rows and columns and cells are defined. Subsequently recognized tables are validated checking if they are “permissible” tables.

In [118], table detection is mapped to a problem of identifying table lines, which is addressed by taking advantage of the “sparse-line” property (i.e., table rows are sparse in terms of text density). Line types are identified to train labels (e.g., states in a CRF model), whereas orthographic, lexical and layout features are considered for document layout representation. Table lines are labeled based on a manually built-up list of possible starting keywords of table captions. The method in [118] considers coordinate features in locating table boundaries, which are essential to the table structure decomposition. This method has been enhanced in [117] to deal with multi-column tables. However, accuracy in line reconstruction is affected by the set-up of a number of threshold parameters. The idea of emphasizing coordinate features in table boundary detection is denitely right.

5.3 PDF Wrapping

Most of the Web wrapping approaches could easily be extended to deal with PDF documents preliminarily converted into a semistructured format (e.g., HTML). Examples of PDF Wrapping approaches that use this techniques are described in papers [51, 23, 194, 195]. But converting a PDF document into another format is usually an imprecise task, often in terms of both formatting and structure layouts. For example, (part of) some text lines could be misaligned, bold texts could be reconstructed as two identical text blocks roughly overlapped to each other, and so forth.

Few systems that do not need a conversion in other formats has been propose. The theoretical foundations of a method for wrapping PDF documents were proposed in [65], followed by their publication [62] with experimental results and their work [66]. In this method, a wrapper is dened as a set of hierarchically nested fuzzy logic conditions.

Aumann et al. [11] describe a system for visual information extraction that use a set of training documents for recognizing rules that allow to extract textual fields like (section) titles, authors, publication lists. This approach is unable to recognize and extract tables but it is interesting because proposes a well defined PDF handling methods.

Authors of papers [87] have addressed the PDF wrapping problem by enhancing the Lixto system to deal with PDF documents. Their method borrows some techniques from computer vision for page segmentation, i.e., dividing a page into atomic blocks that are expected to contain one logical entity in the document (e.g., paragraph, heading, table cell). This task exploits the presence of visual clues on the page and can be performed by adopting a top-down (worst) or bottom-up (best) segmentation approach. Text blocks are then represented as nodes in an attributed relational graph and are connected to each other if spatial relations (e.g., adjacency, alignment) hold. Finally, an error-tolerant algorithm for relational subgraph matching is used to find text blocks that are recognized as somehow similar. This graph-based matching method has been very recently improved in [84], where a subgraph isomorphism is employed into an interactive wrapping system.

Semantic Information Extraction

Today, there is increasing demand for semantic services that allow users to enrich and interlink existing information with ontological meta-annotations. The envisioned semantic Web [83, 22] that would offer this functionality is a slow development process. On the other hand, Web IE systems have started to add support for semantic labeling and even automatic detection of such relational information from labeled and unlabeled sources [187, 46]. Ontology-based information extraction has recently emerged as a subfield of information extraction. Here, ontologies play a crucial role in the information extraction process. Ontologies are used during the information extraction process and/or as output format. A lot of work concerning the use of ontology for extracting meaningful information from HTML Web documents has been proposed in literature. One of the first work in this area is [60]. Recently many approaches appeared, some of the most relevant are described in [6, 125, 40, 123]. A sub area of ontology named extraction ontologies propose methods to use ontologies in IE [163, 190, 154]. At the best of our knowledge no existing approach that deals with the problem of extracting information from PDF documents by using ontologies has already been proposed. In this chapter we only discuss about the typical architecture of ontology-based extraction system. For an almost complete and very recent survey, we refer to [189] paper.

6.1 Ontology-based Information Extraction System

As defined in a recent survey [189], an *ontology-based information extraction system* (OBIE system) is a system that processes unstructured or semi-structured natural language text through a mechanism guided by ontologies to extract certain types of information and presents the output using ontologies. A limitation of this OBIE systems is that they consider only flat text.

OBIE systems are characterized by the following factors:

- Process unstructured or semi-structured documents. Classical OBIE systems are able to consider unstructured (flat text files) or semi-structured documents (Web pages, e.g. Wikipedia pages) and they process the contained natural language text. Images, diagrams or videos, 2-dimensional objects are not considered as input. Input was limited to natural language text [189]. In [143] we extend the ability to process also PDF document format. We consider not only flat text but also 2-dimensional objects and images.
- Present the output using ontologies. The important characteristic that distinguishes OBIE systems from IE systems is the target of recognizing ontologies concepts during the extraction process. In [189] the distinguishing characteristic is that as the o In [111] ontologies are used as input and also as target output. Whereas there exist some OBIE systems that construct the ontology to be used through the IE process itself instead of treating it as an input (e.g. [192]). There exist other systems that use different techniques in order to populate an ontology [179, 47], this task is often known as ontology population.
- Use an IE process guided by an ontology. As defined in [189] in all OBIE systems, the IE process is guided by the ontology to extract things such as classes, properties and instances. This means that no new IE method is invented but an existing one is oriented to identify the components of an ontology. Ontology construction is generally not associated with IE, it can be seen as an important step in the OBIE process [15].

Query Languages

The most common approach to data extraction is to executed by using a special-purpose Language. Such languages are often hidden to users whereas extraction systems allow for visually specifying (by means a graphical user interface) which are the information to extract and which condition are required. Interesting languages falls in the areas of visual languages aimed at manipulating visual information, and query languages for multimedia database and presentation.

7.1 Logic-based Query Languages

For extracting information, several logic-based (in particular Datalog-based) languages have been defined [74, 167, 168, 155, 103, 26]. In order to extract information from Web an *information extraction function* is defined. It is an unary query which takes a Web document represented as labeled unranked tree and returns a subset of its nodes. A wrapper is a program which implements one or several such functions, and thereby assigns unary predicates to document tree nodes. Therefore, a wrapper is capable of return a new tree which maintains only relevant nodes and re-labels nodes.

A well founded logic wrapping language is monadic datalog. Monadic datalog is datalog having only *unary intensional predicates* named *unary query*. A unary query is a function that selects a subset of nodes of the input DOM. A logic-based wrapper is a set of information extraction functions represented by a monadic datalog program. Lixto approach [17] is based on a datalog-like language, ELog. Elog core was shown to be expressively equivalent to monadic second-order logic over finite node-labeled trees. Monadic second-order logic (MSO) is a second-order logic where second-order quantifiers range over sets (i.e., unary relations). A *MSO query* is a MSO formula ϕ with one free first-order variable. Let t be a tree and let V be the set of nodes of t , then $\{n \in V | t \models \phi(n)\}$. In [74] was argued that that MSO unary queries over trees are an appropriate expressiveness yardstick for information extraction

functions. MSO over trees is well-understood theory-wise and quite expressive. It has also been used as an expressiveness yardstick for node-selecting XML query languages. By restricting the structures considered to trees, monadic datalog acquires a number of nice properties. XLog [167] is very closed to Elog.

7.2 Spatial Query Languages

Languages aimed at manipulating visual information. Kong et al. presented [102] a formalism that uses the visual appearance of Web pages in order to extract relevant information from them. Such a formalism, named spatial graph grammars (SGGs), extends graph grammars by incorporating spatial notions into language constructs. SGG productions are used to describe parts of Web pages by a graph representation. The representation includes spatial relations of the following types: direction relations that describe an order in the space, topological relations that express neighborhood and incidence, and distance relations such as near and far. The system allows for extracting page contents of interest for the user, but it is quite complex in term of both usability and efficiency.

A recent and interesting work has been presented in [48]. In this approach the extraction of information is driven by the spatial arrangement of the elements in the web page (e.g., spatial relations such as “right” of or “included in”). In order to query the spatial relations the authors have designed a SQL-like query language, namely SRQL (Spatial Relation Query Language), which allows one to write queries based on the visual arrangement of the web page contents in an intuitive way. Moreover, the SRQL can also use further attributes of the web page elements to refine the queries, thus providing a framework where the information extraction can be performed integrating spatial relations, visual attributes, textual content and document structure.

7.3 Multimedia Query Languages

A lot of work is available in the field of query languages for multimedia database and presentation.

A relevant work is due to Lee et al. [108]. In this work authors discuss the problem of storing, querying and manipulating multimedia presentations by using a single database management system software. In this paper authors present a graph data model where multimedia presentations are represented as directed acyclic graphs. Each node of a presentation graph represents a media stream. Edges depict sequential or concurrent playout of streams during the presentation. For querying multimedia presentation graphs, authors present three presentation graph query languages that use temporal operators Next, Connected, and Until, and path formulas. This way path formulas can specify

paths constructed by the nodes of a presentation graph, and content changes among frames of streams and hierarchical relationships between a stream and its contents.

Adah et al [2] present a framework and an algebra for creating and querying multimedia interactive presentations. In the proposed framework an interactive presentation consists of a tree whose nodes represent "non-interactive presentations" (e.g., a sequence of Power Point slides, an HTML page, or a 2-min video clip). Edges from parents to a children corresponds to an interaction (transition between presentation). Authors define a multimedia presentation algebra (MPA) that contains generalizations of select, project and join operations in the relational algebra. The operators allow the querying of interactive presentation databases based on the contents of individual nodes as well as querying based on the presentations tree structure. Existing approaches and languages are specifically aimed at querying presentation databases or describing the layout of web pages . However, these languages are too complex and not suitable for querying Web pages.

Information Extraction: New Approaches

The best way to predict the future is to invent it.
~ Alan Curtis Kay ~

Towards a Spatial and Semantic Information Extraction System

In this chapter is presented a reference architecture of an information extraction system capable of extracting information from presentation-oriented documents by using spatial and semantic document features. Figure 8.1 depicts the high-level architectural view of the reference spatial and semantic information extraction system. Beside standard capabilities of existing IE systems (i.e. the ability to manage natural language text, navigate the Web and create Web wrappers [18] as happens in Web IE systems, exploit knowledge encoded in ontologies as happens in OBIE systems), the system presented in this dissertation exploits the high level semantic relations made available by the spatial arrangement of presentation-oriented documents. As shown in next chapters the ability to exploit spatial features of PODs enables more effective features that improve and simplify wrapper definition and learning tasks. Furthermore, the system is able to manage in unified way large documents having different internal formats, and provides facilities that enable a better interaction with users.

8.1 Architecture and Functionalities

The system takes as input presentation-oriented documents having different internal format (i.e. HTML, PDF) and queries defined by users. It internally represent PODs in unified way and use such a representation for computing queries and learning extraction rules (wrappers). So the possible outputs are: query answers, learned wrappers, information extracted from documents and stored in structured form in a knowledge base that is automatically populated. The high level architecture of the system consists of several linked units that are described in the following.

The *internal document representation builder* creates the spatial representation of PODs used in the system for computing queries and learning extraction rules. This unit exploit layout engines of Web browsers and rendering engines of PDF document visualization tools in order to obtain the

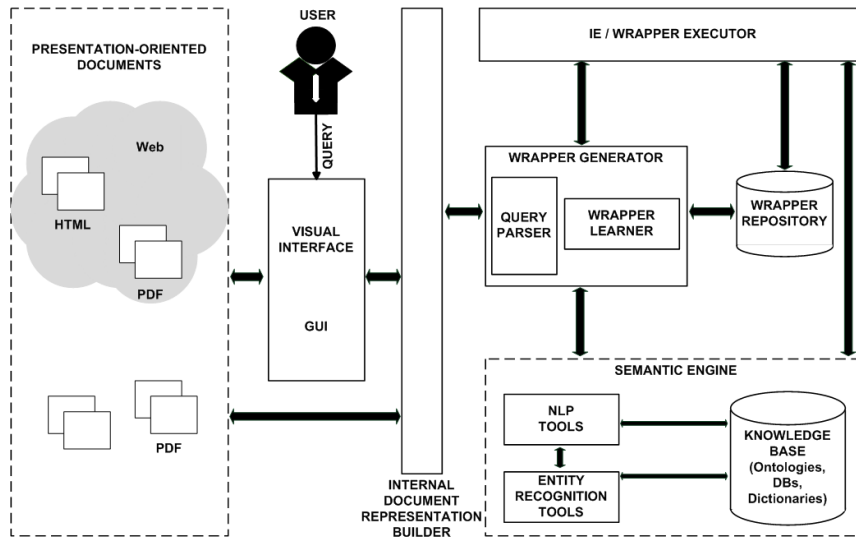


Fig. 8.1. Architecture of an Information Extraction System

spatial positioning of content items in documents. The output of this unit are the 2-Dimensional Flat document representation and the SDOM.

The *wrapper generator* is the unit that supports the user in the query writing process by exploiting the spatial document representation. It generates the wrapper that answer to a specific query written by the user or creates wrappers in unsupervised way. During the query design process this unit supports the user by parsing user queries and displaying on screen the results of the edited query. This way the user can refine the query in order to obtain the desired result. The user can ask this unit to learn automatically wrapper for data records or tables contained in Web pages or PDF documents. Wrapper learning includes the generalization of examples. For instance, an expression that precisely identifies a specific data item, could be generalized replacing some elements by wildcards. So, the result is a generalized wrapper that matches several similar data items on similarly structured pages. If the extraction process requires the navigation of a Web site, navigation actions required for accessing the target page (for example, hyperlink navigation and form filling) can be automatically registered and reproduced during the wrapper execution. Generated wrappers are stored in the *wrapper repository*, this way wrappers can be reused or executed in batch mode.

The *wrapper executor* is the unit that applies wrappers created by the *wrapper generator* to target PODs. It executes Deep Web sites navigation activities registered along with the wrapper, opens PDF documents, etc. The application of the wrapper to target documents consists in the identification

of target objects described in the wrapper, and in the generation of the structured output format.

The *semantic engine* is the unit of the system that enables to answer user queries in semantic way and allows to populate ontologies. When user queries involve concepts stored in an ontology, natural language processing, entity recognition, and semantic analysis of the lexicon are required for computing query results. For example, the WordNet toolkit is widely used for the English language. It groups English words into sets of synonyms (called synsets) and provides semantic relationships between them including a taxonomy. The ontology that is used by the system may be extended or even generated internally by an ontology generator component. Humans may assist the system in the ontology generation process, by applying population rules or by manually editing the ontology. Beside traditional extraction facilities from flat text provided by existing OBIE systems, the proposed information extraction system is able to extract information by taking into account the spatial arrangement of contents in order to identify structures like tables, and also to consider images or other multimedia information.

The *wrapper repository* unit stores wrappers, navigation tasks and their metadata. This way wrapper templates, that could use also domain knowledge, can be reused, updated, and executed in batch mode.

It is noteworthy that because the system can carry out different goals, in a specific wrapping process some architectural components could not be used. Thus, for instance, when a user needs to extract table structures, it is not necessary to exploit the knowledge base; or if a user wants to query specific information in a web page, automatic tasks can help the users but are not mandatory.

The system is supplied by a visual interface that allows the user to: (i) visualize results; (ii) define which data should be extracted from web pages, for instance by selecting the pieces of information that are relevant and by assigning labels to them; (iii) describe how this data should be mapped into a structured format; (iv) improve, in graphical way, the definition of wrappers. The visual interface itself can include several windows, such as: (i) a browser window that renders PODs for the user. Thus, a user can explore the interesting information also using visual features. For instance, s/he is able to individuate in a simple way which is the interesting data region of Deep Web pages; (ii) a parse tree window that shows the HTML parse tree or a tree structure obtained by a document understanding process, of the current POD rendered by the browser. A user can click on a node of the parse tree, and the system immediately highlights the corresponding part of the example page in the browser, and vice versa. Thus, it is more simple for a user to generate, check and correct errors even considering the tree representation. (iii) a control window that allows the user to view and control the overall progress of the wrapper design process and to input textual data (e.g. labels); (iv) a program window that displays the wrapper program constructed so far and allows the

user to further adjust or correct the program, also showing the results in the original visualized document;

In the following chapters of this dissertation approaches, techniques, and algorithms that allows for building units of the system presented above are described in detail.

Document Model for PODs

Presentation-Oriented Documents (PODs) have a very different internal representation. The internal encoding of HTML documents is based on the tree structure, whereas PDF documents are described by a content stream, which contains a sequence of graphical and textual objects located at precise positions within the document pages that express only where contents have to be visualized after rendering. Moreover, the internal document models of PODs are inadequate to support the spatial and/or semantic querying, as described in Sections 2.1 and 2.2. For these reasons, it is very difficult to extract relevant content from such various sources. Querying data from PODs, for purposes such as information extraction, requires the consideration of tree structures as well as the consideration of spatial relationships between laid out elements. The information contained in the internal structure could, if correctly processed, be combined with spatial information and used to improve the robustness of information extraction and querying of PODs.

This chapter define a data model that allow for exploiting visual cues of presentation-oriented documents. The Spatial Document Object Model (SDOM) is able to describe visual objects arrangement when the input document is (i) without any semi structure, e.g. flat text documents, scanned images, PDF documents; (ii) supplied with some type of structured information, like HTML documents. Firstly, the used spatial algebra used for representing spatial relations is described. The subsequent Section of this chapter presents a 2-dimensional flat document model, which allows for obtaining simply spatial information. Then, the logical structure encoded in HTML documents or that can be obtained by segmenting in bottom up way PDF documents is considered. Finally, the logical structure of PODs is extended by spatial relations and ordering information among elements. The obtained model is thought in order to have a lossless representation, make available spatial and pre-existing internal format information. Data model presented in this chapter were described partially in IJAIT 2009 [143] and in VLDB 2010 [145].

9.1 Qualitative Spatial Relations

This section aims at presenting spatial models and basic ideas which the Information Extractions approaches are founded on. Tables and textual areas recognition by human readers is a visual process based on spatial relationships existing among content items. In such a process alignment among groups of content items is visually observed in order to recognize text areas and tables.

PODs can be seen as a Cartesian plane on which cues are arranged. More complex relations among cues can be recognized reproducing the human behavior also only looking at the visualization areas where such information are placed. In the following we introduce the *minimum bounding rectangle* of any information recognized in a document, and a formal model used to represent relations among such rectangles.

Definition 9.1. Let n be a visual clue of a presentation-oriented document, the minimum bounding rectangle (MBR) of n is the minimal rectangle r that surrounds the contents of n and has sides parallel to the axes (x and y) of the Cartesian plane (see Fig. 9.1). The function $mbr(n)$ returns the rectangle r assigned to the visual cue n . We call r_x and r_y the segments that are obtained as the projection of r on the x -axis and the y -axis respectively. Then, each side of the rectangle is represented by the segments (r_x^-, r_x^+) and (r_y^-, r_y^+) , where r_x^- (resp. r_y^-) denote the infimum on the x -axis (y -axis) and r_x^+ (resp. r_y^+) denote the supremum on the x -axis (y -axis) of the segments r_x and r_y .

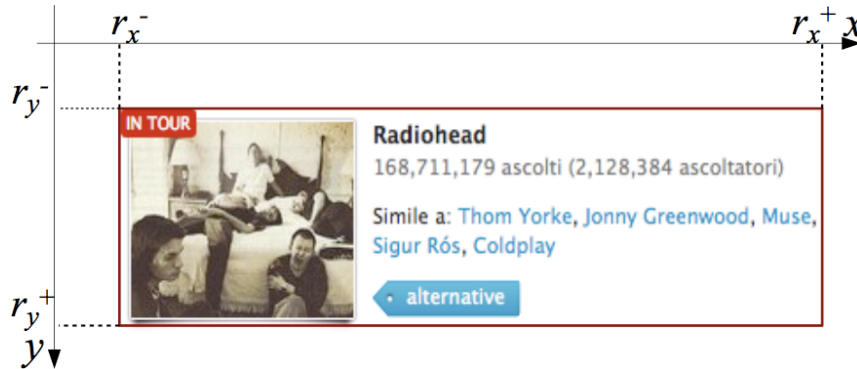


Fig. 9.1. An example of MBR

The function mbr that returns the minimum bounding rectangle is implemented by the layout engine, for instance a PDF visualizer or a browser.

For representing the SDOM we exploit rectangle algebra (RA) [13] as a very fine-grained and expressive model that allows for computations of spatial relations as well as algebraic optimizations. However, the rectangle algebra is

often too fine grained and verbose, for instance to state querying primitives. Therefore we introduce more intuitive models rectangular cardinal relations (RCR) [132] and topological relations [79] for spatial relationships. Moreover, RCRs and topological relations are mapped into RA such that we can choice to store all RA relationships in the SDOM and exploiting the mapping the actual navigation of the SDOM can be performed by means simple axes.

Rectangle Algebra

The rectangle algebra model (RA) has been introduced in [13, 14], which allows for representing all possible relations between rectangles the sides of which are parallel to the axes of some orthogonal basis in a 2-dimensional Euclidean space. RA is a straightforward extension of the standard model for temporal reasoning, the *Interval Algebra* (IA) [8], to the 2-dimensional case. IA models the relative position between pairs of segments by a set of 13 atomic relations (R_{int}), namely *before* (b), *meet* (m), *overlap* (o), *start* (s), *during* (d), *finish* (f), together with theirs inverses $\{bi, mi, oi, si, di, fi\}$ and the relation *equal* (e). For a pictorial representation of IA relations see Table 9.1. Let s and s_1 be two segments the IA relation $s b s_1$ represents that the segment s is preceded by the segment s_1 .

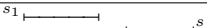
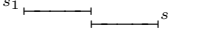
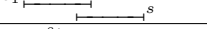
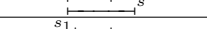
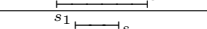
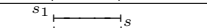

Relation	Symbol	Meaning	Inverse
before	b		bi
meets	m		mi
overlaps	o		oi
starts	s		si
during	d		di
finish	f		fi
equals	e		e

Table 9.1. IA Relations.

Similarly, the domain considered in RA is the set of rectangles REC whose sides are parallel to the axes (x and y) of a Cartesian plane, as happens for MBRs in Definition 9.1. RA models the relative spatial relations between rectangles in REC . Let a and b be two rectangles a *Basic* RA relation between them is a pair (ρ_x, ρ_y) of IA relations in R_{int} : $a (\rho_x, \rho_y) b$. Such a relation holds iff the IA relations $a_x \rho_x b_x$ and $a_y \rho_y b_y$ hold for segments that are obtained as projections of rectangle sides along x (i.e. a_x, b_x) and y (i.e. a_y, b_y) respectively. The expressiveness of RA covers the modeling of all qualitative spatial relations between two MBRs. For a pictorial representation of RA relations see Fig. 9.2. The set of possible Basic RA is called R_{rec} and it is given combining R_{int} relation on x and y -axes. Thus $R_{rec} = R_{int}^2$ contains $13^2 = 169$ elements as shown in Figure 9.2. Furthermore, since the RA model

is an algebra, it holds many important properties (e.g. invertibility) that allow optimized query evaluation.

Since for each visual cue of a presentation oriented document is associated a MBR by means of the function mbr , the Basic RA relations between couple of nodes can be used to model spatial relations in the SDOM.

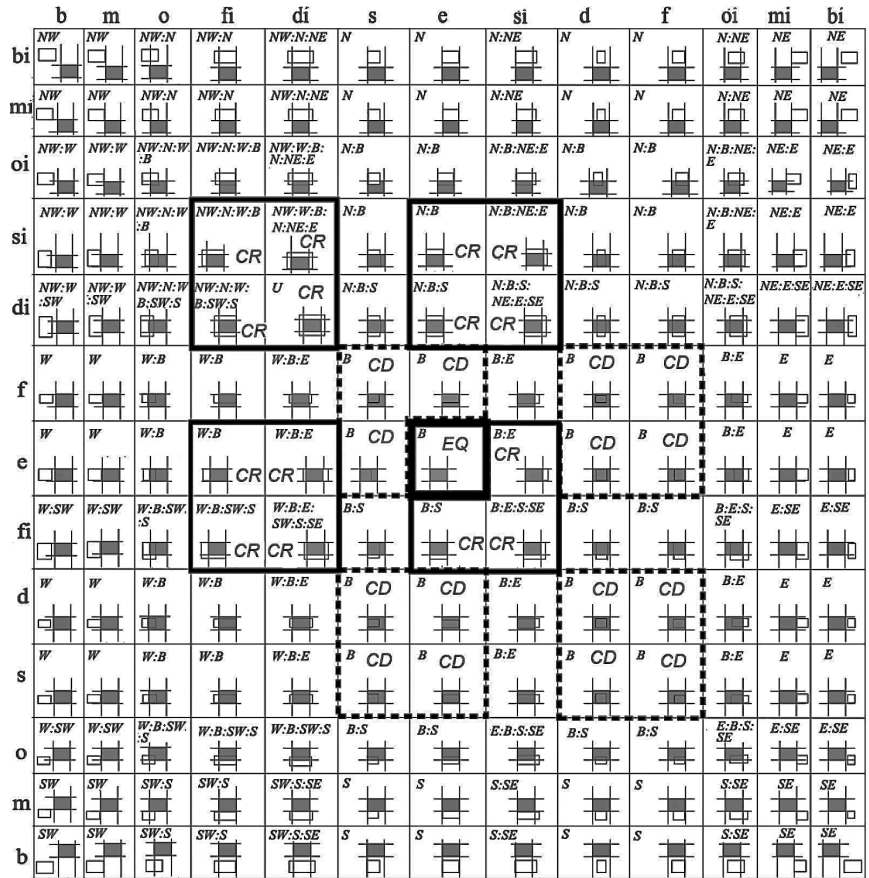


Fig. 9.2. Pictorial Representation of the RA relations

Rectangular Cardinal Relations

In the Rectangular Cardinal Relations model (RCR), proposed in [132], directional relations hold between rectangles in REC . Let r_1 and r_2 be two rectangles in REC . RCRs are expressed by analyzing the 9 regions (*cardinal tiles*) formed, as shown in Figure 9.3, by the projections of the sides

of the *reference rectangle* r_1 . We are interested in how the reference rectangle r_1 is related to r_2 , which we call *resulting rectangle*, through a cardinal relation. By considering cardinal tiles *atomic* RCRs are traditionally expressed by means of the 9 symbols contained in the following set $R_{card}^A = \{B, S, SW, W, NW, N, NE, E, SE\}$. The symbol B denotes the *central tile* and the relation *belongs to*. Other symbols in R_{card}^A correspond to the *peripheral tiles* and denote the eight RCRs *South*, *SouthWest*, *West*, *NorthWest*, *North*, *NorthEast*, *East*, and *SouthEast*.

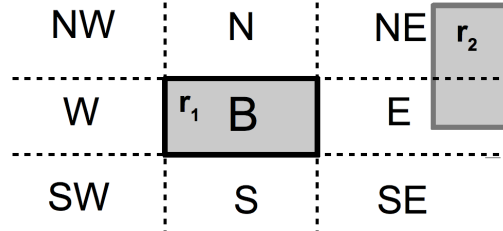


Fig. 9.3. Cardinal tiles

Given $r_1, r_2 \in REC$ and $R \in R_{card}^A$, the expression $r_1 R r_2$ means that r_2 lies entirely on the tile R of r_1 . When a rectangle r_2 holds contemporary in more tiles w.r.t. a reference rectangle r_1 , then we call the relation that links r_2 to r_1 a *multi-tile relation* and it is represented by $R_1 : \dots : R_k$ where R_1, \dots, R_k are atomic RCRs. Atomic and multi-tile relations are called *Basic Relations*. For instance, by considering Figure 9.3 the RCR between the reference rectangle r_1 and the resulting rectangle r_2 is expressed as $r_1 E:NE r_2$. This rectangular cardinal relation means that the rectangle r_2 lies on east and north-east of the rectangle r_1 . In the RCR model, the set R_{card} of all basic RCRs between any couple of MBRs contains 36 elements [132] as shown in Table 9.2.

Disjunctions of Rectangular Cardinal Relations

In qualitative spatial reasoning based on the RCR model it is common to express uncertain spatial relations, between couples of rectangles in REC , by means of disjunctions of basic RCRs that generate the power set $2^{R_{card}}$. Let $\delta = \{\delta_1, \dots, \delta_n\}$ be an element in $2^{R_{card}}$ we write the corresponding disjunctive RCR as $\delta = \delta_1 | \dots | \delta_n$. So let $r_1, r_2 \in REC$, $r_1 \delta r_2$ is satisfied iff $r_1 \delta_1 r_2 \vee r_1 \delta_2 r_2 \vee \dots \vee r_1 \delta_n r_2$.

Example 9.2. By considering Figure 9.4a and 9.4b the relation $r_1 N: NW:NE | B:W:NW:N:NE:E r_2$ satisfies both the spatial arrangements because the basic relation $r_1 N: NW:NE r_2$ holds for Figure 9.4a and $r_1 B:W:NW:N:NE:E r_2$ holds for Figure 9.4b.

	RCR :=		RA-relations
1	B := { d, s, f, e }	×	{ d, s, f, e }
2	S := { d, s, f, e }	×	{ m, b }
3	N := { d, s, f, e }	×	{ mi, bi }
4	W := { m, b }	×	{ d, s, f, e }
5	E := { mi, bi }	×	{ d, s, f, e }
6	NW := { m, b }	×	{ mi, bi }
7	NE := { mi, bi }	×	{ mi, bi }
8	SW := { m, b }	×	{ m, b }
9	SE := { mi, bi }	×	{ m, b }
10	B:S := { d, s, f, e }	×	{ fi, o }
11	B:N := { d, s, f, e }	×	{ si, oi }
12	B:W := { fi, o }	×	{ d, s, f, e }
13	B:E := { si, oi }	×	{ d, s, f, e }
14	S:SW := { fi, o }	×	{ m, b }
15	S:NW := { fi, o }	×	{ mi, bi }
16	S:SE := { si, oi }	×	{ m, b }
17	N:NE := { si, oi }	×	{ mi, bi }
18	W:SW := { m, b }	×	{ fi, o }
19	W:NW := { m, b }	×	{ si, oi }
20	E:SE := { mi, bi }	×	{ fi, o }
21	E:NE := { mi, bi }	×	{ si, oi }
22	B:W:E := { di }	×	{ d, s, f, e }
23	B:S:N := { d, s, f, e }	×	{ di }
24	S:SW:SE := { di }	×	{ m, b }
25	NW:N:NE := { di }	×	{ mi, bi }
26	SW:W:NW := { m, b }	×	{ di }
27	NE:E:SE := { mi, bi }	×	{ di }
28	B:S:E:SE := { si, oi }	×	{ fi, o }
29	B:N:NE:E := { si, oi }	×	{ si, oi }
30	B:S:SW:W := { fi, o }	×	{ fi, o }
31	B:W:NW:N := { fi, o }	×	{ si, oi }
32	B:S:SW:W:NW:N := { fi, o }	×	{ di }
33	B:S:SE:E:NE:N := { si, oi }	×	{ di }
34	B:S:SW:W:E:SE := { di }	×	{ fi, o }
35	B:W:NW:N:NE:E := { di }	×	{ si, oi }
36	U := {(di,di)}		

Table 9.2. RCRs as cartesian products of RA Relations

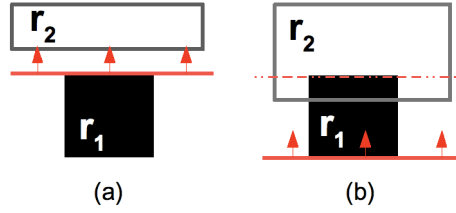


Fig. 9.4. (a) r_1 N:NW:NE r_2 . (b) r_1 B:W:NW:N:NE:E r_2 .

Topological Relations

Qualitative topological relations among objects in a 2-dimensional Euclidian space are described in the region connection calculus (RCC) model and its variants. Two of the most known RCC models are RCC5, which is depicted in Figure 9.5 and RCC8 [157]. In this paper we introduce 3 topological relations, inspired by the RCC5 model, namely: *contained* (CD), *container* (CR), and *equivalent* (EQ). We denote the set of topological relation as $R_{topo} = \{CD, CR, EQ\}$. Let r_1 and r_2 two MBRs, r_1 CD r_2 , r_1 CR r_2 , and r_1 EQ r_2 means that the rectangle r_2 is contained in r_1 , is the container of r_1 , and is equivalent to r_1 respectively.

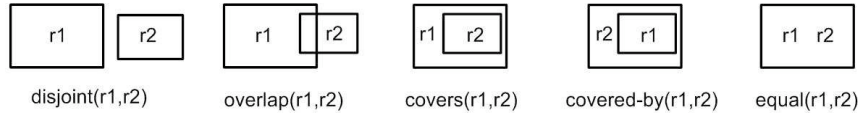


Fig. 9.5. RCC5 model

Example 9.3. By considering Figure 9.5, we have r_1 CD r_2 (RCC5 relation *covers*), r_1 CR r_2 (RCC5 relation *covered_by*), and r_1 EQ r_2 (RCC5 relation *equal*).

It is worthwhile nothing that the three topological relations in R_{topo} satisfy the *transitive closure* property. Let r_1, r_2, r_3 be three rectangles and $\tau \in R_{topo}$ be the considered spatial relation, if $r_1 \tau r_2$ and $r_2 \tau r_3$ hold, then $r_1 \tau r_3$ is also satisfied.

	Top. rel. :=	RA-relations	
37	EQ := { (e,e) }		
38	CR := { { fi, di, e, si } }	×	{ si, di, e, fi } - {(e,e)}
39	CD := { { d, s, f, e } }	×	{ d, s, f, e } - {(e,e)}

Table 9.3. Topological relations mapped into RA Relations

Mapping

The correspondence between RCRs and RA relations and between topological and RA relations can be computed by means of the mapping function defined as follows.

Definition 9.4. *Let R_{card} , R_{topo} and R_{rec} be the set of basic (atomic and conjunctive) RCRs, topological and RA relations respectively, then the mapping function $\mu : 2^{R_{card}} \cup R_{topo} \rightarrow 2^{R_{rec}}$ that maps RCRs and topological relations in terms of RA relations is:*

$$\mu(R) = \begin{cases} \text{as in table 9.2} & \text{if } R \in R_{card} \\ \text{as in table 9.3} & \text{if } R \in R_{topo} \\ \mu(R_1) \cup \dots \cup \mu(R_k) & \text{if } R = R_1 | \dots | R_k \end{cases}$$

Example 9.5. The relation NW reported in table 9.2 has $\mu(NW) = \{m, b\} \times \{mi, bi\}$ i.e. $\mu(NW) = \{(m, mi), (m, b), (b, mi), (b, bi)\}$.

A pictorial representation of the Mapping Function μ (Def. 9.4) is shown in Fig. 9.2.

9.2 2-Dimensional Flat Representation

Broadly speaking, the main idea which this model is based on is that POD can be considered as a Cartesian plane on which are arranged MBRs, which contains *content items* (CIs).

A CI is an atomic piece of content (i.e. an images, an alphanumeric string written with unique font features, a graphical or typographical element like a line etc.) visualized in a rectangular area of the plane.

So the spatial document model of a POD consists in a set of CIs which are formally defined as follows.

Definition 9.6. *Let T be a set of type names arranged in a taxonomy, a content item is a 3-tuple of the form:*

$$CI = \langle \tau, \sigma, \alpha \rangle$$

where:

- $\tau \in T$ is the type of the content item (such as image, text, number, percentage, currency, etc).
- σ is the value of the content item (such as a string, the URL of an image, etc).
- $\alpha = \langle r_x^-, r_y^-, r_x^+, r_y^+ \rangle$ is an MBR that surround the contents σ (see Figure 9.6), where (r_x^-, r_y^-) and (r_x^+, r_y^+) are the Cartesian coordinates of the two opposite vertices (top-left and bottom-right).

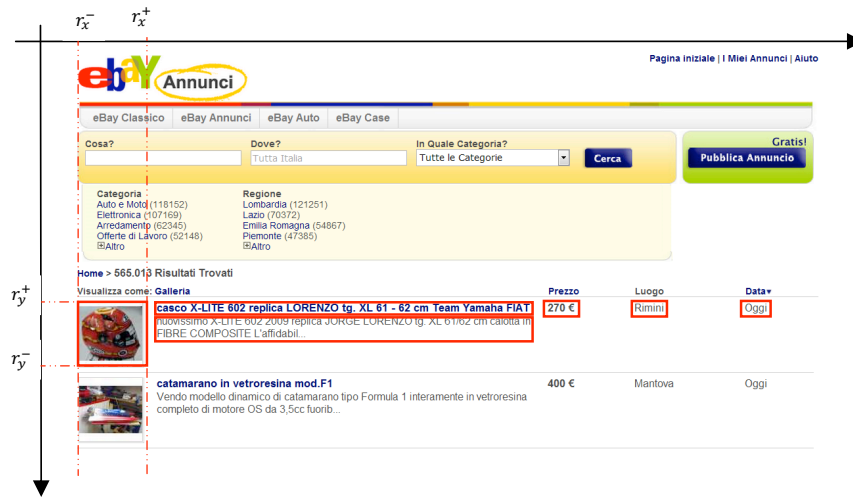


Fig. 9.6. A page of Ebay Web Site with highlighted some CIs

Figures 9.6 and 2.6 show some CIs bounded by rectangles having sides parallel to the axes of the Cartesian plane (that is, MBRs).

Presentation-oriented documents are visualized by layout engines embedded in web browsers (e.g. Netscape Gecko for Firefox¹) and PDF visualizers (e.g. Adobe Acrobat Reader²). The main feature of layout engines and PDF visualizers is that they consider the area of the screen aimed at visualizing a document, as a 2-dimensional Cartesian plane on which they arrange CIs. In particular, *Layout Engines* of Web Browsers assign to each node in the DOM a *visualization area* on the plane computed by applying rendering rules. Rendering rules take into account the DOM structure and *cascade style sheets* (CSS) [180] that equip the web document. It is worthwhile noting that the appearance of a Web page depends from visualization conditions, such as browser windows size and font dimension. Therefore, the model is produced in function of the rendering condition. Whereas, PDF visualizers interpret parameters encoded in the PDF stream [4] and assign a visualization area on the plane to each content item. In a PDF document, visualization areas are defined during document editing and encoded in the *PDF Stream* (i.e. sequence of instructions in the PDF language) when the document is stored [4]. So, different PDF visualizers interpret the *PDF Stream* in a standard way and give to documents the same appearance.

In order to compute the 2-D flat model, we exploit the coordinates of the MBRs assigned by the layout engine to each visualized DOM node and parameters encoded in the PDF stream, for HTML and PDF format respec-

¹ <https://developer.mozilla.org/en/Gecko>

² <http://www.adobe.com/products/reader/>

tively. For better identify CIs of PDF documents, the strategy described in the Section 10.1 can be used. Besides spatial coordinate, to support the composability of the complex full-text query, strings contained in the CIs, may be processed. Such processing consists in: (i) tokenization of strings contained in final CIs, and the acquisition, for each token, of its position and linguistic features (i.e. POS-Tag and lemma) by means of a NLP tool; (ii) the matching of dictionary pattern on obtained strings; (iii) the extraction of presentation features (e.g. font style, type, color, size, link) for each CI, directly acquired by using node attributes or PDF metadata.

The 2-D flat model enable to reproduce the human behavior. In fact, tables and textual areas recognition by human readers is a visual process based on spatial relationships existing among content items. In such a process alignment among groups of content items is visually observed in order to recognize text areas and tables. Section 10.1 shows as the 2-D model flat can be exploited in order to implement a system, named PDF-TREX, the allows for analyzing, recognizing and extracting information from PDF files. PDF-TREX starts from an initial set of basic elements, and by using a heuristic algorithm that considers only their spatial features, aligns and groups them in bottom-up way in order to identify text areas and tabular arrangements of information. Moreover, Section 11.2 and Chapters 12, rather than concentrate only on the documents physical structure, consider also the textual content. So, they enables to express 2-dimensional composition rules that permit to recognize and extract information and even ontology objects contained in PODs also when information is arranged in tabular form.

9.3 Spatial Document Model – SDOM

Querying data from Web documents, such as from other presentation-oriented formats such as PDF, requires a versatile machinery to abstract from presentation structure into conceptual relationships. Germane to such abstraction is the transformation into structural languages such as XML, indeed quite often such XML languages are used as presentation formats, e.g. HTML 5. Based on these structural languages, one may use query formalisms such as XPath and XQuery in order to turn presentation structure into meaningful relational or logical representations.

While, the internal representation of HTML documents in terms of DOM represents an abstraction from presentation structure of documents into conceptual relations, the intrinsic print/visual oriented nature of PDF encoding poses many issues in defining “ad hoc” IE approaches. Techniques of document understanding can be applied in order to detects a tree structure that facilitate further processing, and model an equivalent structure like for HTML documents. The obtained tree will be correspond to a HTML DOM, where root and elements node will be corresponds to *composite CIs* (CCIs) that is defined in the following, whereas leaf nodes are simple CIs.

Definition 9.7. A Composite Content Item (CCI) is a 2-tuple of the form:

$$CCI = \langle \Gamma, \alpha \rangle$$

where:

- Γ is a non empty set of CIs.
- $\alpha = \langle r_x^-, r_y^-, r_x^+, r_y^+ \rangle$ is an MBR such that:
 - $r_x^- = \min(r_{x_\gamma}^- | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
 - $r_y^- = \min(r_{y_\gamma}^- | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
 - $r_x^+ = \max(r_{x_\gamma}^+ | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$
 - $r_y^+ = \max(r_{y_\gamma}^+ | \gamma \in \Gamma \wedge \gamma = (\tau, \sigma, r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+))$

In this section we do not bind the spatial document model to any document understanding techniques for the following motivations:

- The tree representation does not allow us to express all the relationships we might need to locate wrapping instances, so we want not include a priori a subset of relations in this section. Nevertheless, in next chapter, we will propose a simple tree structure for PDF documents, which may help users to query documents in intuitive way and enable automatic table recognition process.
- It is crucial that the document understanding process detects the desired structures. However, document understanding is inherently an inaccurate process, and this approach lacks robustness.

As shown in Section 2.1, the tree-based structures of XML are often not convenient and sometimes even not expressive enough in order to represent the presentation layout and its corresponding conceptual relationships. The layout often indicates the semantics of data items but arising on real-world web pages, such spatial arrangements are rarely explicit and frequently hidden in complex nestings of layout elements. Therefore, we enrich the tree document model by visual/spatial relations among visualized elements.

The Spatial DOM (SDOM) is a XML document model (DOM) represented by a node labeled sibling ordered tree [112, 76, 124] enriched by: (i) basic spatial *rectangle algebra* (RA) relations [14] existing between pairs of nodes visualized on screen.(ii) spatial total orders among nodes.

Definition 9.8. SDOM is a node labeled sibling ordered tree [76, 112, 124] enriched by RA relations. It is described by the following 7-tuple:

$$SDOM = \langle V, root, (label_a)_{a \in \Sigma}, R_{\Downarrow}, R_{\Rightarrow}, A, f_s \rangle$$

where:

- V is the set of labeled DOM nodes. $V = V_v \cup V_{nv}$, where V_v is the set of nodes visualized on screen, and V_{nv} is the set of nodes that are not visualized.

- *root* is a unary relation, which contains the root of the tree.
- *label* is a unary relation, such that $\text{label}_a(n)$ is true if *n* is labeled *a* in the tree.
- R_{\downarrow} is the firstchild relation. Let *n* and *n'* be two nodes in V , $nR_{\downarrow}n'$ holds iff *n'* is the first child of *n*.
- R_{\Rightarrow} is the nextsibling relation. Let *n* and *n'* be two nodes in V , $nR_{\Rightarrow}n'$ holds iff *n'* is the next sibling of *n*.
- $A \subseteq V_v \times V_v$ is the set of arcs that represent spatial relations between pairs of nodes visualized on screen.
- Let R_{rec} be the set of RA relations, $f_s : A \rightarrow R_{rec}$ is the function that assigns to each element in A a RA relation in R_{rec} . So, let *n* and *n'* be two nodes in V_v , we have $a = (n, n') \in A$ holds iff $mbr(n) f_s(a) mbr(n')$.

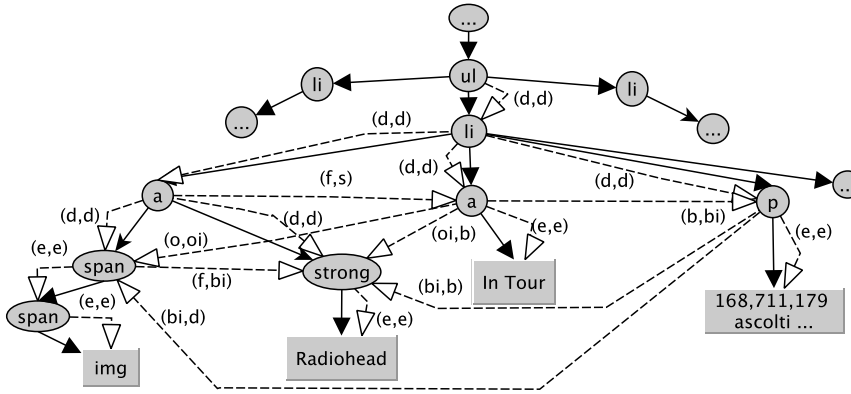


Fig. 9.7. A SDOM Fragment of the Web Page Portion in Fig. 11.1

Fig. 9.7 depicts the SDOM for a portion in Fig. 9.1 of web page shown in Fig. 2.2. Solid arrows represent the classical DOM tree structure that is equivalent to the DOM depicted in Fig. 2.3. Dashed labeled arcs represent the RA relations between pairs of nodes. For instance, the arc (d, d) between nodes *ul* and *li* represents the RA relation $mbr(ul)(d, d)mbr(li)$, i.e. both IA relations $mbr(ul)_x d mbr(li)_x$ and $mbr(ul)_y d mbr(li)_y$ hold, and means that the rectangle $mbr(li)$ is spatially contained in the rectangle $mbr(ul)$. In order to improve readability of the figure, spatial relations are represented only for a subset of nodes.

The SDOM model provides orders among nodes defined as follows.

Definition 9.9. Document order relations for a SDOM are:

- the document total order $<_{doc}$, (as already defined in Def. 2.2). Let $u, w \in V$ be two nodes, then $u <_{doc} w$ iff the opening tag of u precedes the opening tag of w in the (well-formed version of the) document.
- the document directional total orders \leq_{\uparrow} (from south), \leq_{\rightarrow} (from west), \leq_{\downarrow} (from north), and \leq_{\leftarrow} (from east). Let $u, w \in V_v$ be two nodes with MBRs $a = mbr(u)$ and $b = mbr(w)$ respectively, we have $u < w$ (or $u = w$) iff:
 - $a_y^- < b_y^-$ ($a_y^- = b_y^-$) for \leq_{\uparrow} ,
 - $a_x^- < b_x^-$ ($a_x^- = b_x^-$) for \leq_{\rightarrow} ,
 - $a_y^+ > b_y^+$ ($a_y^+ = b_y^+$) for \leq_{\downarrow} ,
 - $a_x^+ > b_x^+$ ($a_x^+ = b_x^+$) for \leq_{\leftarrow} .
- the containment partial order \leq_t . Let $u, w \in V_v$ be two nodes with $a = mbr(u)$ and $b = mbr(w)$ respectively, we have $u \leq w$ iff $a_x^- \leq b_x^- \leq b_x^+ \leq a_x^+$ and $a_y^- \leq b_y^- \leq b_y^+ \leq a_y^+$. In particular, $u = w$ if $a_x^- = b_x^-$, $b_x^+ = a_x^+$, $a_y^- = b_y^-$, and $b_y^+ = a_y^+$.

In the following we introduce the concept of layering adopted for computing spatial order among nodes of a given node set.

Definition 9.10. Let $\Gamma \subseteq V_v$ be a set of SDOM nodes, for each spatial order \leq_z , where $\leq_z \in \{\leq_{\uparrow}, \leq_{\rightarrow}, \leq_{\downarrow}, \leq_{\leftarrow}, \leq_t\}$, the spatial layering of nodes in Γ w.r.t. \leq_z is an ordered list of non-empty lists $L_{\leq_z}(\Gamma) = \{l_{1_{\leq_z}}, \dots, l_{h_{\leq_z}}\}$, where let u, w be two nodes in Γ :

- if $u = w$ w.r.t. \leq_z , then $u, w \in l_{i_{\leq_z}}$ for some $i_{\leq_z} \in \{1, \dots, h_{\leq_z}\}$.
- if $u < w$ w.r.t. \leq_z , then $u \in l_{i_{\leq_z}}$ and $w \in l_{j_{\leq_z}}$ where i_{\leq_z} and j_{\leq_z} are the smallest numbers such that $i_{\leq_z}, j_{\leq_z} \in \{1, \dots, h_{\leq_z}\}$ and $i_{\leq_z} < j_{\leq_z}$.

Example 9.11. Considering Fig. 5 and the spatial order \leq_{\uparrow} , we obtain the layering $L_{\leq_{\uparrow}}(\{n_1, \dots, n_6\}) = \{l_{1_{\leq_{\uparrow}}}, l_{2_{\leq_{\uparrow}}}, l_{3_{\leq_{\uparrow}}}\}$ where $l_{1_{\leq_{\uparrow}}} = \{n_1\}$, $l_{2_{\leq_{\uparrow}}} = \{n_2, n_4, n_6\}$, and $l_{3_{\leq_{\uparrow}}} = \{n_3, n_5\}$. The layering corresponds to the following directional total order from south: $n_1 \leq_{\uparrow} n_2 \leq_{\uparrow} n_4 \leq_{\uparrow} n_6 \leq_{\uparrow} n_3 \leq_{\uparrow} n_5$.

In order to facilitate the definition of SXPath language semantics we introduce the concepts of *spatial index function* and *last index function* as follows.

Definition 9.12. Let $\Gamma \subseteq V_v$ be a set of SDOM nodes, and let $L_{\leq_z}(\Gamma) = \{l_{1_{\leq_z}}, \dots, l_{h_{\leq_z}}\}$ be the spatial layering of nodes in Γ w.r.t. \leq_z , where $\leq_z \in \{\leq_{\uparrow}, \leq_{\rightarrow}, \leq_{\downarrow}, \leq_{\leftarrow}, \leq_t\}$. The spatial index function $pid_{\leq_z}(u, \Gamma)$ returns the index of the layer which the node u belongs to (where 1 is the smallest index). The last directional index function $plast_{\leq_z}(\Gamma)$ returns the index h_{\leq_z} .

The SDOM makes machine processable the high level semantics expressed in the spatial arrangement of content items of presentation-oriented documents. Then SDOM allows the definition of XML query languages that consider tree structures as well as spatial relationship.

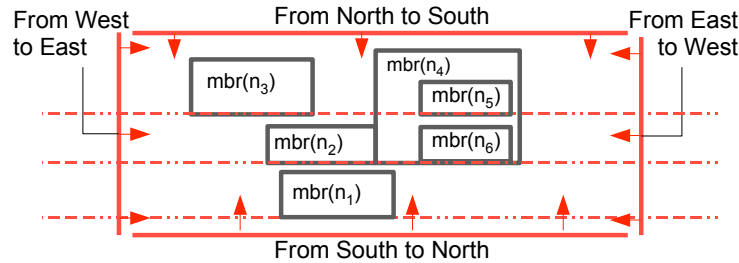


Fig. 9.8. Ordering directions

SDOM Implementation

In this section it is briefly described the processing strategies, depicted in Figure 9.9, that has been implemented for obtaining the SDOM of presentation-oriented documents. First, the classical hierarchical representation (DOM) augmented with spatial coordinates, i.e. for each DOM node its *MBR* is created. We call this structure *Positional DOM* (PDOM). Then, the *SDOM Builder* computes RA relations nodes and defines nodes directional orders giving a structured representation to visual cues planned by web designers, so the SDOM is obtained. The subprocess that allow us to obtain the PDOM is related to the type of the internal structure of the presentation oriented document.

For documents encoded in HTML the *SDOM Builder* takes as input the DOM and the *visualization areas* respectively produced by the *HTML parser* and the *Layout Engine* embedded in the browser. The *Layout Engine* of the browser for creating the visual representation uses the DOM structure that encode content items to display, and the *Style Sheets* that encode the visual appearance of content items planned by web designers. So, visualization areas hold the coordinates that define the MBRs in which DOM nodes are visualized on screen. It is worthwhile noting that for web documents the *SDOM Builder* produces the SDOM real time at each rendering of the page (e.g. browser window resizing). The appearance of a web page, in fact, depends from visualization conditions (e.g. screen resolution, dimensions of the visualization area assigned to the browser on the screen). Thus, for each web page, each layout engine (i.e. the web browser), and in each visualization condition there is a unique corresponding SDOM.

In a PDF document, visualization areas are defined during document editing and encoded in the *PDF Stream* (i.e. sequence of instructions in the PDF language) when the document is stored [4]. So, different PDF visualizers interpret the *PDF Stream* in a standard way and give to documents the same appearance. The *PDF Harvester* analyzes the PDF stream and identifies content items and their visualization parameters. (i.e. coordinates of the rectangles in which content items are visualized on screen). Then, a technique of *document analysis and understanding* is apply to obtain an ad-hoc PDOM.

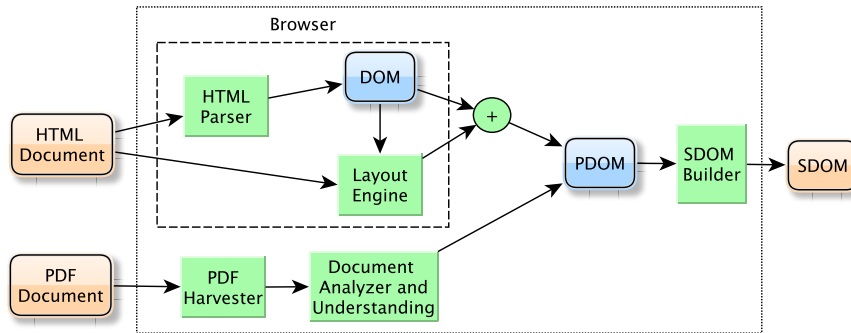


Fig. 9.9. The SDOM building process

The implemented classes accomplish the goal of maintain the complete compatibility with W3C specification. In fact, classes implements interfaces (indicated with “I” as start symbol in the Figure 9.10) that agree with W3C specifications. A PDOM is a Document (DOM) having its nodes of PositionalNode type, which subsumes the types HTMLPositionalNode and PDFPositionalNode

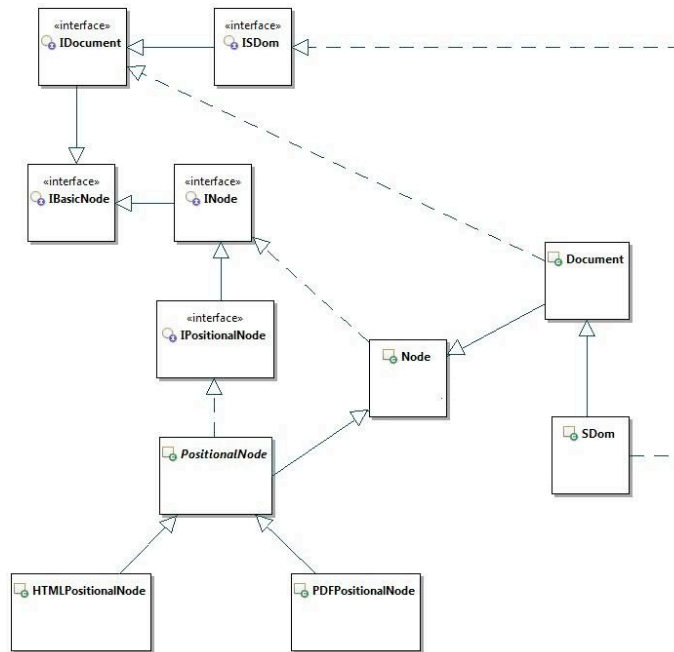


Fig. 9.10. Class Diagram of Document Model Implementation

9.4 Discussion

In this section have been presented the 2-Dimensional Flat Model and the Spatial Document Model (SDOM). The first model represents a POD as a set of 2-Dimensional content items. The second model extends the first one by means of rectangle algebra relations between pairs of content items. Furthermore, the SDOM extends the standard DOM model allowing to use navigation facilities characteristic for this kind of document representation. The main feature of such models is that they allow for representing in a unified way presentation-oriented documents having different internal encodings. This feature enable the definition of languages and automatic wrapper learning techniques (presented in next chapters) that are able to query PODs and learn wrappers from them independently from the internal encodings. In particular, the ability to represent the geometric structure of PODs enables to define query languages more intuitive for human users and wrapper induction techniques that exploit the spatial arrangement of content items for inferring tables and groups of data records presented in PODs. Motivations that draw the definition of such document models is that designers of PODs tend to take the properties of rendering engines for more important than a sound conceptual modeling of document elements. In fact, designers use the spatial arrangement of content items as a way for better expressing the semantics of presentation-oriented document contents. Hence, from a conceptual point of view presented models make machine processable the high level semantics expressed in the spatial arrangement of content items of PODs. The extension of the spatial document model by further semantic annotations of content items, and further semantic relation between pairs of content items, could allow for defining object extraction methods for PODs independent from the structure of documents. Roughly speaking, the exploitation of spatial and semantic relation represented in an extended version of the SDOM could allow to define novel approaches (that go beyond those described in this thesis) aimed at recognizing a single or multiple instances of a given class of objects in documents having very different layouts.

Automatic Information Extraction

This Chapter describes two approaches that allow for recognizing and extracting in automatic way valuable information and visual structures (like tables and data records) from PODs.

The first approach, implemented in the *PDF-REX* system, allow for performing document analysis of PDF documents. Given page of a PDF documents, it performs bottom-up segmentation on the text and construct a tree DOM structure that enable document understanding steps. In particular, the heuristic technique for table recognition and extraction from PDF documents is presented. The system uses only document-generic knowledge and is designed to work on an any PDF documents. Moreover, the approach proposed works directly on the geometric structure of PDF documents, thus avoiding to miss relevant information. Empirical evaluation has been performed on a dataset of 100 PDF documents containing complex tables mixed with text. A detailed analysis of type of errors is performed and results are compared to the state of art system [88] that performs automatically table extraction from PDF files. This approach was first published at ICDAR 2009 [140].

The second approach is aimed at automatically extracting information from Deep Web sites. The SDOM representation is exploited in order to automatically recognize repetitive data records in Deep Web pages, and learn the wrapper. In particular, such a process exploits the internal structure as well as the spatial regularities and allow for obtaining better results compared to the existing approaches. The first prototype has been implemented in a system named *SILA* (Spatial Instance Learning Approach) [146]. Empirical evaluation on 100 Deep Web sites (which consider also web pages used in DEPTA [198] and STAVIES [150] papers) has been performed for testing its performances and comparing results with two systems presented in literature, which also are aimed at automatically recognizing repetitive records. This approach was first presented as a ICAR-CNR technical report in 2010 [146].

10.1 PDF-TREX: Understanding PDF Documents

This section describes the technique used in PDF-TREX system to create a DOM representation for PDF documents. The PDF-TREX approach considers a PDF document as a Cartesian plane on which are placed MBRs (see Definition 9.1). The PDF-TREX approach reproduces human behavior. Tables and textual areas recognition by human readers, in fact, is a visual process based on spatial relationships existing among content elements. In such a process alignment among groups of content elements is visually observed in order to recognize text areas and tables. So, for instance, a group of characters can be interpreted as a word when it is separated from other groups of characters by a given space, a group of words constitutes a text line or a text block depending from spatial words distribution. Moreover, a group of lines or one or more text blocks can constitute a text paragraph or a table depending from their mutual spatial positioning. Similarly, the PDF-TREX approach starts from an initial set of CIs, and by using a heuristic algorithm aligns and groups them in bottom-up way in order to identify text areas and tabular arrangements of information. For instance, in Figure 10.4 are depicted the Cartesian plane and final MBRs obtained from the PDF document shown in Figure 10.6¹. Figure 10.4 also highlights the box $\beta = \langle \text{"$ Tornado Warning"} , 1, 32, 4, 33 \rangle$. It is noteworthy that coordinates assigned to MBRs preserve their relative-positions. An additional output of the approach is a DOM. That is a set of node with associate a specific MBR, which can be organized in hierarchical way. Each node corresponds to: (i) a Content Item (CI) (see Definition 9.6), where it is a leaf node; (ii) a Composite CI (CCI) (see Definition 9.7), where it is the root or an element node.

The PDF-TREX approach is based on an heuristic algorithm, which steps are described in the following:

1. CI harvesting. By accessing the PDF content stream of a document, initial CIs are identified and acquired with their spatial coordinates (MBRs) (see Figure 10.2) and, if available, with presentation features (e.g. font style, type, color, size, link).
2. Space distribution analysis. This is a fundamental step that allows to identify how MBRs are distributed in a page. The algorithm analyzes the horizontal and vertical space distribution among MBRs of a page, and defines horizontal and vertical distance threshold values based also on local features (e.g. blank size depends on used font size).
3. Lines building. In this step the algorithm checks the horizontal coordinates of MBRs and identifies those that can be considered belonging at the same line.
4. Segments building and lines tagging. MBRs belonging to the same line are clustered in order to obtain *segments*, by considering distance thresholds. When a line contains only one segment it is tagged as "text line" if the

¹ Obtained from <http://english.wunderground.com>

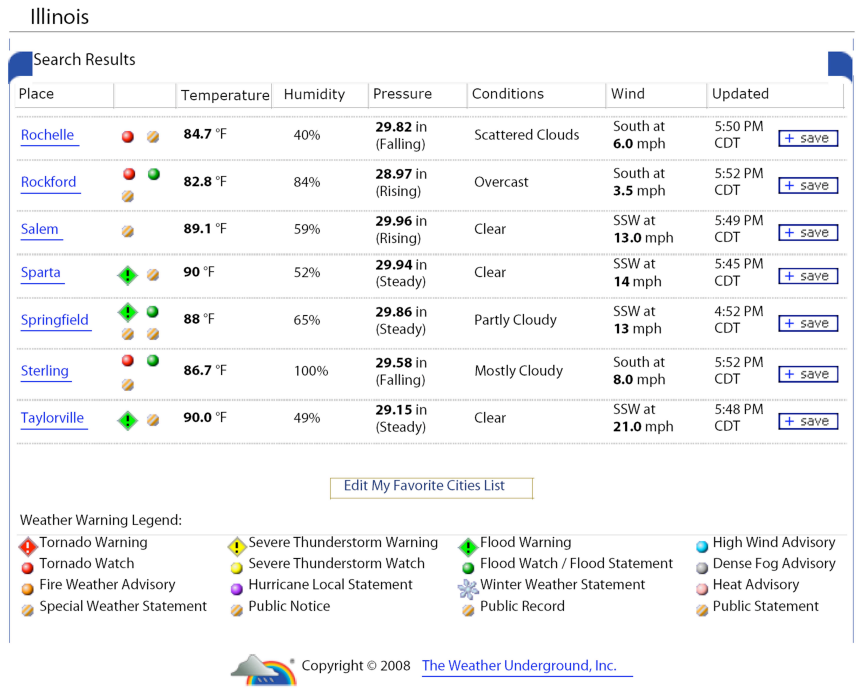


Fig. 10.1. The Input PDF Document

- segment cover more than half line length, as "unknown line" otherwise. When a line contains more than one segment it is tagged as "table line".
5. Areas building. The algorithm analyzes the sequence of lines and identifies possible text and table areas as sequences respectively of text lines and table lines, eventually with inserted unknown lines. When images that cover a significant number of lines are identified an image zone is defined.
 6. Blocks and Rows building. Segments contained in an area are clustered until vertical distance is below the related threshold value. Segments embraced in a cluster are arranged in a blocks (Figure 10.3) which coordinates depend from those of contained segments. Blocks are represented by boxes which strings are obtained by concatenating strings (preserving visualization order) of segments contained in the block. In order to identify table rows, shown in Figure 10.4, the same approach used for constructing lines is applied to blocks.
 7. Columns Building. When segments and blocks belonging to a table area are overlapped in vertical direction, then they are assigned to a table column. When a segment or a block overlaps more than one other segment or block, it is assigned to two different columns. This trick allows to recognize column headers spanning on multiple columns.

- Tables and paragraph building. In this step the algorithm crosses identified table rows and columns for defining the grid of table cells (Figure 10.4). Paragraph are identified merging lines belonging to text areas.

Sterling		86.7 °F	100%	29.58 in (Falling)	Mostly Cloudy	South at 8.0 mph	5:52 PM CDT	+ save
Taylorville		90.0 °F	49%	29.15 in (Steady)	Clear	SSW at 21.0 mph	5:48 PM CDT	+ save

Fig. 10.2. Initial CIs obtained from the input in Fig. 10.1

Sterling		86.7 °F	100%	29.58 in (Falling)	Mostly Cloudy	South at 8.0 mph	5:52 PM CDT	+ save
Taylorville		90.0 °F	49%	29.15 in (Steady)	Clear	SSW at 21.0 mph	5:48 PM CDT	+ save

Fig. 10.3. Clusters obtained from the input in Fig. 10.1

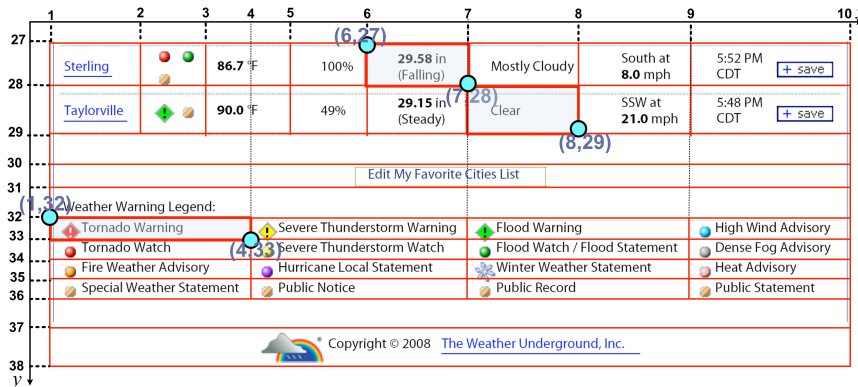


Fig. 10.4. Cartesian plane and final MBRs obtained from the input in Fig. 10.1

PDF-TREX takes a PDF document as input. The PDF document is analyzed in order to construct its 2-dimensional flat representation (see Section 9.2) and its DOM.

Besides taking as input a PDF document, PDF-TREX is able to take dictionaries of regular expressions representing specific named entities, and also to interact with NLP tools. As described in Section 9.2, strings contained in the CIs, may be furthermore processed. Such processing consists in: (i) tokenization of strings contained in final CIs, and the acquisition, for each

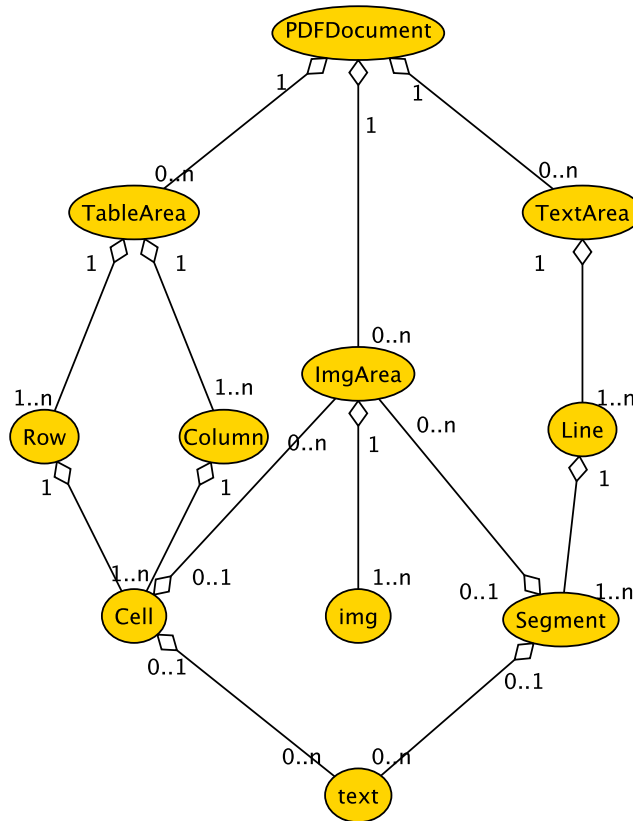


Fig. 10.5. Logical architecture of Types of PDF DOM

token, of its position and linguistic features (i.e. POS-Tag and lemma) by means of a NLP tool; (ii) the matching of dictionary patterns on obtained strings; (iii) the extraction of presentation features (e.g. font style, type, color, size, link) for each CI, directly acquired by using PDF metadata. In this way, complex full-text query is enabled.

Likewise for HTML documents, PDF DOM can be extended by spatial relations and orders in order to create the SDOM, as described in Section 9.3.

In the following the heuristic approach aimed at recognizing and extracting tables, is defined in detail.

10.1.1 Recognizing and Extracting Tables

Tables are widely used in digital documents as an essential structuring component to concisely convey content. A relevant topic in document understanding and analysis is indeed regarded as table detection and reconstruction.

Definition 10.1. A **Table** is data structure to organize the tuples of a relation: $\{\langle value_{ix}, value_{iy}, \dots \rangle, \langle value_{jx}, value_{jy}, \dots \rangle, \dots\}$ in a structured way. ■

A table represents an arrangement of numbers, words, symbols or items of any kind, in columns and rows. Tables are visual oriented arrangements of information widely used in many different domains as a ways to present and communicate complex information to human readers. They allow humans to rapidly and simply read and understand relation among different set of information, to understand trends and the behavior of a given quantity. For example, companies' financial statements contain a lot of tables regarding companies' assets, liabilities, and net equity in a given point in time; scientific documents present experimental results in a synthetic way by using tables; furthermore, in decision support systems tables (generated by OLAP tools), are the principal way to present information needed for decision making.

Tables are an effective way to show values, making comparisons but when represented in unstructured documents, they are well suited for human users but need automatic extraction for store them in structured way. Automatically extracting information contained in tables and storing them in structured machine-readable form is of paramount importance in many application fields. Tables have many different layouts and are mostly contained in semi-structured and unstructured documents having various internal encodings (e.g. HTML, PDF, flat text). More specifically, table contained in document encoded by the portable document format (PDF) are particularly difficult to recognize and extract because such a document format is completely without structures and their internal encoding needs to be interpreted. For these reasons table recognition and extraction is a very challenging problem that poses many issues to researchers and practitioners in defining effective approaches.

As described in Section 5.2 a significant body of work concerning approaches and systems for table recognition and extraction (TRES) is currently available in literature. However, currently there is not a winning approach. An important limitation of TRES, as a "per se" research field, is the lack of available standard datasets that hinders objective and complete comparisons among existing approaches. Furthermore, TRES approaches for PDF documents suffer from the following limitations: (i) there is not a clear definition of exploited document features; (ii) handled objects are somehow high level document structures (e.g. text fragments) and graphical feature but is not clear how they are obtained and combined.

PDF-TRES is an heuristic approach for table recognition and extraction from PDF documents. The scope of the approach is to recognize tables contained in PDF documents as a 2-dimensional grid on a Cartesian plane and extract them as a set of cells equipped by 2-dimensional coordinates. The PDF-TRES approach considers a PDF document as a Cartesian plane on which are placed *content elements* contained in MBRs. The heuristics starts

from an initial set of basic content elements and aligns and groups them, in bottom-up way by considering only their spatial features, in order to identify tabular arrangements of information. So, the approach does not require: (i) linguistic or domain knowledge; (ii) graphical metadata and ruling lines; (iii) predefined table layouts. To better show how the approach works, a running example is considered. The output obtained by PDF-TREX can be serialized in a preferred document format (e.g. excel, XML) or in a structured form (like tuples Datalog). Recognized tables can be further processed for understanding table contents and improving information extraction from PDF documents. For instance, in [139] is described a reasoning method that exploits a logic-based representation of extracted cells and domain knowledge for storing table contents as instances of a knowledge base.

Preliminary Definitions

Let be $\alpha_1 = \langle x_1^t, y_1^t, x_1^b, y_1^b \rangle$ and $\alpha_2 = \langle x_2^t, y_2^t, x_2^b, y_2^b \rangle$ two MBRs, the following definitions hold.

Definition 10.2. α_1 and α_2 are said to be horizontally overlapped when by projecting α_1 from left to right along the Y -axis, α_2 is contained in (or intersect) such a projection. The following inequalities formally describe horizontal overlapping conditions: (i) $y_2^t \leq y_1^t \leq y_2^b \leq y_1^b$; (ii) $y_1^t \leq y_2^t \leq y_1^b \leq y_2^b$; (iii) $y_1^t \leq y_2^t \leq y_2^b \leq y_1^b$; (iv) $y_2^t \leq y_1^t \leq y_1^b \leq y_2^b$. ■

Definition 10.3. Let $\delta = \min((y_1^b - y_1^t), (y_2^b - y_2^t))$ The horizontal overlapping ratio between α_1 and α_2 is respectively one of the following expressions (depending from overlapping conditions presented in definition 3): (i) $\rho_h(\alpha_1, \alpha_2) = \frac{y_2^b - y_1^t}{\delta}$; (ii) $\rho_h(\alpha_1, \alpha_2) = \frac{y_1^b - y_2^t}{\delta}$; (iii) $\rho_h(\alpha_1, \alpha_2) = \frac{y_2^b - y_2^t}{\delta}$; (iv) $\rho_h(\alpha_1, \alpha_2) = \frac{y_1^b - y_1^t}{\delta}$. ■

Definition 10.4. When α_1 and α_2 are horizontally overlapped their horizontal distance is: (i) $D_h(\alpha_1, \alpha_2) = (x_2^t - x_1^b)$, if $x_1^b < x_2^t$; (ii) $D_h(\alpha_1, \alpha_2) = (x_1^t - x_2^b)$, if $x_2^b < x_1^t$. When α_1 and α_2 are not horizontally overlapped $D_h(\alpha_1, \alpha_2) = \infty$. ■

Vertically overlapped elements, vertical overlapping ratio and vertical distance are defined in analogous way.

Heuristic Algorithm

The heuristic algorithm is organized in eight steps *elements harvesting, lines building, lines tagging, areas building, rows building, columns building, tables building* and *tables extraction*. Their behavior is described in the following by using as running example the PDF page² shown in Figure 10.6.

² The page has been obtained from a balance sheet of an Italian company

La composizione del patrimonio netto (articolo 2427 n. 7-bis)

Nel prospetto che segue viene evidenziata la composizione del patrimonio netto, con specifico riferimento alla possibilità di utilizzazione e alla distribuibilità delle singole poste nonché alla loro utilizzazione negli esercizi precedenti all'anno fiscale chiuso al 31/12/2004.

	Riserva			Totale
	Capitale sociale	legale	straordinaria	
Alla chiusura dell'esercizio precedente	301.600	34.179	757.409	1.277.968
Destinazione del risultato dell'esercizio:				
- altre destinazioni		9.280	175.500	(184.780)
Altre variazioni:				
- distribuzione dividendi			(275.000)	
Risultato dell'esercizio corrente				210.046
Alla chiusura dell'esercizio corrente	301.600	43.459	657.909	1.213.014

Il valore iscritto in bilancio presenta un decremento, rispetto al precedente esercizio, di 23.776 euro; la composizione e la movimentazione dei debiti sono descritte nella tabella sottostante:

Composizione	31/12/2003	variazione	31/12/2004
debiti verso banche	247.829	-85.641	162.188
debiti verso altri finanziatori	201.746	-27.649	174.097
altri debiti	69.545	-1.059	68.486
debiti verso fornitori	81.971	-23.200	58.771
debiti tributari	38.045	45.491	83.536
debiti verso personale e istituti di previdenza	202.568	68.281	270.849
Totale	841.704	-23.777	817.927

Fig. 10.6. The Running Example Page

Elements Harvesting.

In the harvesting step initial *basic* that have the form described in Definition Defintion 9.6, are identified by accessing the PDF document, and then acquired and stored. In basic CIs: (i) the string σ is a characters sequence that do not contain blank chars; (ii) coordinates of MBRs are assigned, starting from positional information contained in the PDF document, so that there are no couples of basic CIs that have both horizontal and vertical overlap-

ping ratio greater than 0. In Figure 10.7 are depicted basic CIs (graphically represented by rectangular box) acquired from a section of the input page. The figure also shows a visualization area (which box is highlighted by bold border) and the Cartesian plane in which its coordinates are defined. In the harvesting step *horizontal* and *vertical distance threshold values* (hT and vT) are computed. These values, used in next steps, are evaluated by analyzing white space sizes and horizontal and vertical distance distributions among visualization areas.

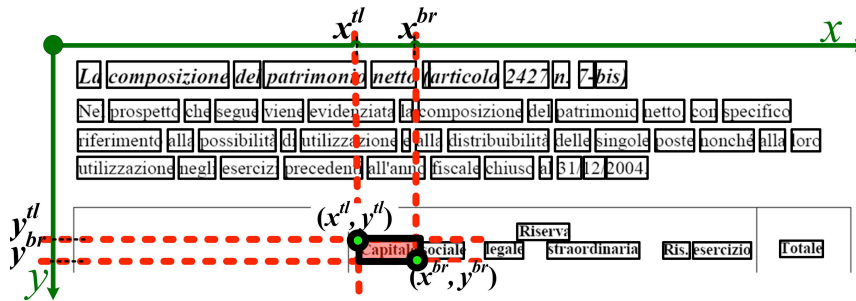


Fig. 10.7. Basic Content Items

Lines Building.

In this step *lines*, having the form described in the following definition, are built.

Definition 10.5. A line is a 2-tuple $\lambda = \langle E, \alpha \rangle$ in which: (i) E is a set of CIs; (ii) horizontal coordinates of α correspond respectively with the minimum and the maximum horizontal coordinates of a page; (iii) vertical coordinates of α are assigned so that do not exist lines horizontally overlapped. ■

Basic CIs are assigned to a line when their horizontal overlapping ratio is over a given threshold³. It is noteworthy that CIs assigned to lines are obtained from basic CIs by adapting their vertical coordinates to those of the line that contain them. Figure 10.8 depicts lines (highlighted by dashed border) obtained in a section of the running example page.

Segments Building and Lines Tagging.

Vertical sequences of lines can generate either text paragraphs or tables. The lines tagging step assigns to each line a tag: *text line* (TxL), *table line* (TbL), *unknown line* (UnL). In order to tag lines, CIs contained in a line are grouped in *segments* having the form defined in the following.

³ In experiments horizontal overlapping ratio has been set to 50%.

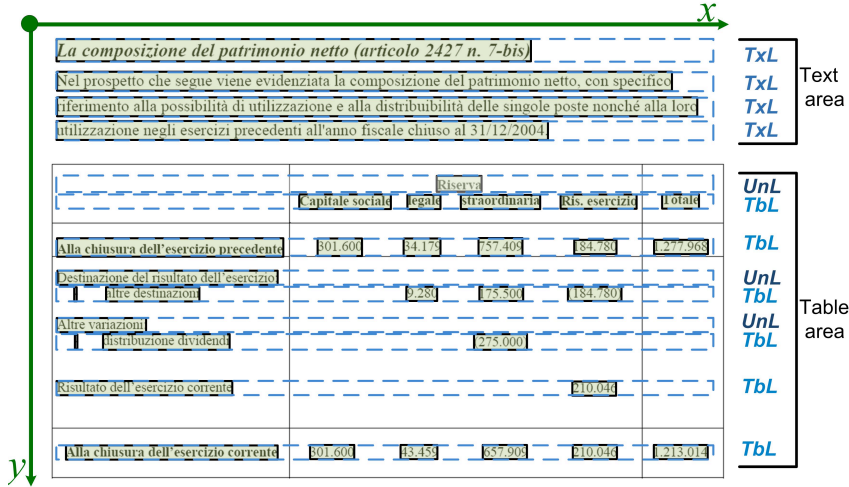


Fig. 10.8. Elements, Segments, Lines, Text and Table Areas

Definition 10.6. A segment is a 2-tuple $\theta = \langle E, \alpha \rangle$ where: (i) $E = \{\varepsilon_i \mid \varepsilon_i = \langle \sigma_i, \alpha_i \rangle \wedge i = 1, \dots, n \wedge \alpha_i = \langle x_i^t, y_i^t, x_i^b, y_i^b \rangle\}$ is a set of CIs; (ii) coordinates of α are: $x^t = \min_{i=1}^n x_i^t$, $y^t = \min_{i=1}^n y_i^t$, $x^b = \max_{i=1}^n x_i^b$, $y^b = \max_{i=1}^n y_i^b$. ■

To construct segments an agglomerative hierarchical clustering algorithm [97] is used. The algorithm works line by line. Initially each CI of a line is assigned to a cluster, then the algorithm agglomerates clusters until horizontal distance is lower than hT . Final clusters represent segments.

A line that contains only one segment is tagged either as *text line* when the segment spans over half horizontal line length, or as *unknown line* otherwise. Whereas, a line is tagged as *table line* when it contains more than one segment. Figure 10.8 depicts segments (highlighted by rectangles having gray background) and line tags obtained in a section of the running example page.

Table Areas Building.

This step aims at analyzing sequences of lines in order to identify table areas defined as shown in the following.

Definition 10.7. A table area is a 2-tuple $T_a = \langle L, \alpha \rangle$, where: (i) L is an ordered list of consecutive lines $L = \{l_i \mid l_i = \langle E_i, \alpha_i \rangle \wedge i = 1, \dots, n \wedge \alpha_i = \langle x_i^t, y_i^t, x_i^b, y_i^b \rangle\}$ tagged only as table lines or unknown lines; (ii) vertical coordinates of α are: $y^t = \min_{i=1}^n y_i^t$, $y^b = \max_{i=1}^n y_i^b$ (iii) horizontal coordinates (x^t and x^b) of α are set by using horizontal coordinates of a MBR of any line in L . ■

Block and Row Building.

Row building step aims at constructing table rows defined as shown in the following.

Definition 10.8. A row is a 2-tuple $r = \langle L, \alpha \rangle$ where: (i) L is a subset of contiguous lines contained in a table area; (ii) horizontal coordinates of α are set by using horizontal coordinates of related table area; (iii) vertical coordinates of α are assigned so that all rows of a table area are contiguous. ■

In order to recognize lines belonging to a same table row, *blocks*, defined as shown in the following, are used.

Definition 10. A *block* is a 2-tuple $\beta = \langle S, \alpha \rangle$ in which: (i) $S = \{s_i \mid s_i = \langle E, \alpha_i \rangle \wedge i = 1, \dots, n \wedge \alpha_i = \langle x_i^t, y_i^t, x_i^b, y_i^b \rangle\}$ is a set of segments; (ii) coordinates of α are: $x^t = \min_{i=1}^n x_i^t$, $y^t = \min_{i=1}^n y_i^t$, $x^b = \max_{i=1}^n x_i^b$, $y^b = \max_{i=1}^n y_i^b$.

To construct blocks the same agglomerative hierarchical clustering algorithm is adopted for constructing segments is used. In this case, segments are the initial clusters, the algorithm agglomerates these clusters until vertical distance is lower than vT . Final clusters represent needed blocks. Normally each line constitutes a row, but when there is a block that embraces segments of a set of consecutive lines and only one line in the set is tagged as *TbL* whereas the others are tagged as *UnL*, the set of line is grouped in a single row. This behavior is very useful for recognizing rows which headers span on multiple lines as a unique logical structure. In Figure 10.9 rows obtained in the running example page (highlighted by dotted lines) are shown.

Column Building.

Column building step aims at creating table columns defined as shown in the following.

Definition 10.9. A column is a 2-tuple $c = \langle S, \alpha \rangle$ in which: (i) S is a set of segments; (ii) horizontal coordinates of α are assigned so that all columns of a table area are contiguous; (iii) vertical coordinates of α are set by using vertical coordinates of related table area. ■

Columns are built by using vertical overlapping ratio among segments and distances among columns contained in table areas. Segments are assigned to the same column when they are vertically overlapped. When a segment overlaps more columns it represents a column headers spanning on multiple columns. When there are columns composed by only one segment (for example column headers not overlapped with column values) horizontal distances are considered. When the distance between such a column and a consecutive column containing more than one segment is below the horizontal threshold hT the two columns are merged in a new single column. Figure 10.9 depicts (by using dotted lines) columns obtained in the running example table areas.

Table Building.

This step generates final tables as sets of 2-dimensional cells defined as follows.

Definition 10.10. A cell is a 2-tuple $c = (\sigma_c, \alpha)$ where: (i) α is obtained by crossing a row and a column; (ii) let E be the set of elements embraced in α , σ_c is the string obtained by concatenating strings contained in E preserving visualization order from left to right and top to bottom. ■

Figure 10.9 shows final cells grid. It is worthwhile noting that the heuristic algorithm is able to recognize cells containing multi line row headers, null values and column headers spanning on multi column as highlighted in Figure 10.9 by using bold border.

Extraction.

This step produces the output of the PDF-TREX approach by serializing in XML table cells recognized in previous step. The output can be further processed for fitting table layouts, understanding table contents, enabling information extraction and document annotation.

```

<TABLE>
  <ROWS>
    <ROW>
      <COORDINATE x_tl=..., y_tl=..., x_br=..., y_br=... \>
      ...
    </ROW>
  </ROWS>
  <COLUMNS>
  ...
  <\COLUMNS>
  <CELLS>
  ...
  <\CELLS>
</TABLE>

```

10.1.2 Experiments**Evaluation Measures**

To measure the goodness of data extracted have been used Precision (P) and Recall (R), two widely used metrics for evaluating the correctness of a pattern recognition algorithm. So, Recall measures the ability of extract relevant material. Precision measures the ability to not extract non relevant material. They can be defined formally on 2X2 contingency table shown in Table 10.1.

Formally, Precision and Recall can be defined as follows:

	Capitale sociale	Riserva legale	Riserva straordinaria	Ris. esercizio	Totale
Alla chiusura dell'esercizio precedente	301.600	34.179	757.409	184.780	1.277.968
Destinazione del risultato dell'esercizio - altre destinazioni		9.280	175.500	(184.780)	
Altre variazioni: - distribuzione dividendi			(275.000)		
Risultato dell'esercizio corrente				210.046	
Alla chiusura dell'esercizio corrente	301.600	43.459	657.909	210.046	1.213.014

Composizione	31/12/2003	variazione	31/12/2004
debiti verso banche	247.829	-85.641	162.188
debiti verso altri finanziatori	201.746	-27.649	174.097
altri debiti	69.545	-1.059	68.486
debiti verso fornitori	81.971	-23.200	58.771
debiti tributari	38.045	45.491	83.536
debiti verso personale e istituti di previdenza	202.568	68.281	270.849
Totale	841.704	-23.777	817.927

Fig. 10.9. Blocks, Rows, Columns and the Final Cells Grid

		Conditions	
		True	False
Prediction	True	TP (true positive)	FP (false positive)
	False	FN (false negative)	TN (truenegative)

Table 10.1. Precision and Recall

$$P = \frac{\#correctlyRetrievedDataItems}{\#RetrievedDataItems} = \frac{TP}{TP + FP}$$

$$R = \frac{\#correctlyRetrievedDataItems}{\#dataItemsInDataSet} = \frac{TP}{TP + FN}$$

Often, there is an inverse relationship between precision and recall, where it is possible to increase one at the cost of reducing the other. Thus, also in this dissertation, both are combined into a single measure, such as the *F-measure* (F), which is the weighted harmonic mean of precision and recall.

$$F = \frac{2PR}{P + R}$$

Experimental Results

PDF-TREX performances are evaluated by using a dataset composed by 100 pages containing tables mix to text, coming from different domains and written in different languages (e.g. companies' financial statements, economic reports). Experiments, carried out on a dataset composed of tables contained in documents coming from different domains, shows that the approach is well performing in recognizing table cells. This dataset ⁴ aimed at contributing to the definition of standard datasets in the TREX field. Indeed, it allowed to perform comparative experiments with a state-of-art tool. In the paper [86] Hassan compared its system with our old version on partial dataset (70 instead of 100 documents). The Table 10.2 reports the precision, recall and F-measure calculated for two version of the presented PDF-TREX system, and compares their results with the Hassan [88] system. The old version of the PDF-TREX approach (v 0.1) considered a table as a perfect matrix, and tried to duplicate cell value in presence of spanning columns; instead in the new version of PDF-TREX (v 0.2), this procedure has been replaced with a more simple, but more effective step necessary to recognize the spanning column.

0	Tables			Cells		
	P	R	F-Measure	P	R	F-Measure
PDF-TREX v0.2	63.78%	96.73%	76.87%	90.41%	99.64%	94.80%
PDF-TREX v0.1	60.15%	95.48%	73.80%	83.66%	98.08%	90.30%
Hassan-tool	63.68%	88.07%	73.92%	84.35%	93.74%	88.80%

Table 10.2. Comparison of precision and recall results for both table areas and cells

⁴ Dataset available at <http://staff.icar.cnr.it/ruffolo/trex/dataset.zip>

Documents have been analyzed by hand in order to identify table areas and table cells existing within them. Then results automatically obtained by the systems have been compared to those manually computed by using precision and recall measures. Considering the choices made by Hassan in paper [86], in order to compute results, the average of the total numbers of detected cells directly over the complete dataset is considered. With this method, documents containing more information (more table areas/cells) were also given more weighting in the final result. The table shows that the new version of PDF-TREX (v 0.2) is better performing and both hither precision and recall.

Evaluation Method

This relatively simple model that consider TP, FP, TN, FN have to consider various types of errors that can occur. Therefore, in this section how the errors are evaluated is described. This methods takes in account the choices done by Hassan in the paper [86] and try to better explain some error conditions and evaluation formula.

Cells errors

In a table we can identify two types of cells: cell containing data and empty cell that does not contain data. These cell, if recognized correctly, are classified as *Found Correctly Data* and *Found Correctly Blank* respectively.

Splitting Error. Such type of error occurs when a cell is wrongly split into multiple cells. We have two different forms of splitting error:

- A cell is split but not its content, so this kind of split add new empty cells. We obtain a *split full* cell that is the cell that contains the data, and additional empty cells that are classified as *void blank*. Figures 10.10(b) show an example of full split obtained taken as input the Figure 10.10(a): a new whole column is created, but the data within them are not split. The cells highlighted in green are classified as **split full**, while Cells in blue are classified as **void blank**.
- A cell and also its content are split. In this case we call the original cell as *split data*, while the additional cell, which also contains a data portion are classified as *void data*.
- A cell spanning several rows or several column is wrongly split. In this case, the splitting is classified in according to cell contents, as shown in previous cases. Figure 10.11(b) show the output obtained from the real table in Figure 10.11(a) which contains horizontal and vertical spanning cells. **split full** and **void blank** related to the horizontal spanning cell are highlighted in green and in blue respectively. While, **split data** and **void data** related to the vertical spanning cell are highlighted in purple and in orange respectively.

Descrizione	Entro 12 mesi	Oltre 12 mesi	Oltre 5 anni	Totale
Verso Clienti	5.670	0	0	5.670
Crediti tributari	14.471	0	0	14.471
Verso altri	9.750	0	0	9.750
Totale crediti	29.891	0	0	29.891

(a)

Descrizione	Entro 12 mesi	Oltre 12 mesi	Oltre 5 anno	Totale
Verso Clienti	5.670	0	0	5.670
Crediti tributari	14.471	0	0	14.471
Verso altri	9.750	0	0	9.750
Totale crediti	29.891	0	0	29.891

(b)

Fig. 10.10. Full Split Error: (a) Input table; (b) output table.

Parameter	Site 7		Site 8		Site 9		Site 13	
	Sampling Days		Sampling Days		Sampling Days		Sampling Days	
	"Wet"	"Dry"	"Wet"	"Dry"	"Wet"	"Dry"	"Wet"	"Dry"
Ammonia (mg/L)	0.143	0.115	0.643	0.939	0.723	0.957	0.108	0.103
Suspended Solids (mg/L)	95	23	25	12	40	22	9	10
Faecal Coliforms (MPN/100 mL)	197	107	116	2	104	39	421	50

(a)

Parameter	Site7		Site 8		Site 9		Site 13	
	Sampling	Days	Sampling days	Sampling days	Sampling days	Sampling Days	Sampling Days	
	"Wet"	"Dry"	"Wet"	"Dry"	"Wet"	"Dry"	"Wet"	"Dry"
Ammonia (mg/L)	0.143	0.115	0.643	0.939	0.723	0.957	0.108	0.103
Suspended Solids (mg/L)	95	23	25	12	40	22	9	10
Faecal Coliforms (MPN/100 mL)	197	107	116	2	104	39	421	50

(b)

Fig. 10.11. Split Spanning Error: (a) Input table; (b) output table.

Merging Error. Such type of error occurs when two or more cells are wrongly *merged*. The merging error can take place for a pair of cells (see Figures 10.12(a) and 10.12(b) that represent the input and output table respectively), or also for entire rows or columns, as shown in Figures 10.13(b) w.r.t. the input table in Figure 10.13(b). cells, as shown in Figures 10.12(a),10.12(b) where the merging error has been highlighted in green.

Sometimes, merging and splitting errors occurs together. The horizontal error has greater priority then the vertical errors. For instance, Figure 10.14(b), which refers to input Figure 10.14(a), shows a split error of the first and fourth column (horizontal splitting), and a merge error of the third row with the fourth (vertical merging). In this case, split has greater priority.

Table errors

In the following possible errors that can occur in the recognition process of the entire table are classified. A table recognized correctly will be named as

Descrizione	Entro 12 mesi	Oltre 12 mesi	Oltre 5 anni	Totale
Verso Clienti	5.670	0	0	5.670
Crediti tributari	14.471	0	0	14.471
Verso altri	9.750	0	0	9.750
Totale crediti	29.891	0	0	29.891

(a)

Descrizione	Entro 12 mesi	Oltre 12 mesi	Oltre 5 anni	Totale
Verso Clienti	5.670	0	0	5.670
Crediti tributari	14.471	0	0	14.471
Verso altri	9.750	0	0	9.750
Totale crediti	29.891	0	0	29.891

(b)

Fig. 10.12. Full Split Error: (a) Input table; (b) output table.

Stati Uniti	961,9	1.008,4	-46,5	1.105,3	1.173,0	-67,7	1.062,7	1.183,2	-120,5	1.291,6	1.428,2	-136,6
Canada	211,5	219,5	-8,0	239,1	237,3	1,8	223,8	228,5	-4,7	275,6	270,4	5,2
Giappone	2.045,1	1.598,0	447,1	1.758,2	1.312,1	446,1	1.576,9	1.078,4	498,5	1.591,4	1.074,3	517,1
Europa (3)	5.025,7	5.218,3	-192,6	5.789,3	6.064,1	-184,8	6.274,0	6.224,2	49,8	7.026,3	7.017,8	8,5
di cui: Germania	1.346,9	1.345,1	1,8	1.630,3	1.638,2	-7,9	1.792,8	1.642,3	150,5	2.083,9	1.885,5	198,4
..... Francia	903,6	968,7	-65,1	1.021,7	1.060,1	-38,4	1.010,7	1.116,3	-105,6	913,6	932,5	-18,9
..... Regno Unito	478,8	528,1	-49,3	559,8	632,3	-72,5	640,7	678,1	-37,4	775,3	833,1	-57,8
..... Italia	419,0	416,8	2,2	443,1	434,0	9,1	382,7	402,3	-19,6	389,9	405,8	-15,9
..... Svizzera	708,4	706,0	2,4	846,1	836,5	9,6	964,0	964,0	0,0	1.170,9	1.171,2	-0,3
Altri	539,0	522,7	16,3	626,7	515,1	111,6	471,6	452,8	18,8	667,7	675,8	-8,1
Totale	8.783,2	8.566,9	216,3	9.518,6	9.301,6	217,0	9.609,0	9.167,1	441,9	10.852,6	10.543,1	309,5

(a)

Stati Uniti	961,9	1.008,4	-46,5	1.105,3	1.173,0	-67,7	1.062,7	1.183,2	-120,5	1.291,6	1.428,2	-136,6
Canada	211,5	219,5	-8,0	239,1	237,3	1,8	223,8	228,5	-4,7	275,6	270,4	5,2
Giappone	2.045,1	1.598,0	447,1	1.758,2	1.312,1	446,1	1.576,9	1.078,4	498,5	1.591,4	1.074,3	517,1
Europa (3)	5.025,7	5.218,3	-192,6	5.789,3	6.064,1	-184,8	6.274,0	6.224,2	49,8	7.026,3	7.017,8	8,5
di cui: Germania	1.346,9	1.345,1	1,8	1.630,3	1.638,2	-7,9	1.792,8	1.642,3	150,5	2.083,9	1.885,5	198,4
..... Francia	903,6	968,7	-65,1	1.021,7	1.060,1	-38,4	1.010,7	1.116,3	-105,6	913,6	932,5	-18,9
..... Regno Unito	478,8	528,1	-49,3	559,8	632,3	-72,5	640,7	678,1	-37,4	775,3	833,1	-57,8
..... Italia	419,0	416,8	2,2	443,1	434,0	9,1	382,7	402,3	-19,6	389,9	405,8	-15,9
..... Svizzera	708,4	706,0	2,4	846,1	836,5	9,6	964,0	964,0	0,0	1.170,9	1.171,2	-0,3
Altri	539,0	522,7	16,3	626,7	515,1	111,6	471,6	452,8	18,8	667,7	675,8	-8,1
Totale	8.783,2	8.566,9	216,3	9.518,6	9.301,6	217,0	9.609,0	9.167,1	441,9	10.852,6	10.543,1	309,5

(b)

Fig. 10.13. Whole Rows and Multi-columns Merging Error: (a) Input table; (b) output table.

Found Correctly.

Additional lines and columns. This type of error corresponds to add rows or columns outside of the input table. In this case the table is classified as *Merged Into Surroundings*, while the additional cells are classified as *Extra Data* if the cell contains data, or as *Extra Blank* if the additional cell is empty. The Figure 10.15(b) shows an example where the table in Figure 10.15(a) is classified as *merged into surroundings* because a row is added, and the cells highlighted in red are classified as *extra data*.

Wrongly data added to boundary cells. This error occur when data is added to a cell along the edge of the table, but not extra cells are added to the table). The table is considered *Found Correctly*, but the additional cells are classified as *Merged External*. Figure 10.16(b) show an example where inside

Descrizione	Entro 12 mesi	Oltre 12 mesi	Oltre 5 anni	Totale
Verso Clienti	5.670	0	0	5.670
Crediti tributari	14.471	0	0	14.471
Verso altri	9.750	0	0	9.750
Totale crediti	29.891	0	0	29.891

(a)

Descrizione		Entro 12 mesi	Oltre 12 mesi	Oltre 5 anni		Totale
Verso	Cliente	5.670	0	0	0	5.670
Crediti	tributari	14.471	0	0	0	14.471
Verso	Altri	9.750	0	0	0	9.750
Totale	crediti	29.891	0	0	0	29.891

legenda
void data
split data
split full
void blank
merged

(b)

Fig. 10.14. Splitting and Merging Errors occur in a same table: (a) Input table; (b) output table.

DESCRIZIONE	IMPORTO AL 31/12/2003
Depositi cauzionali	2.625,00
Partecipazioni	34.143,00
Titoli di stato	27.186,00

(a)

DESCRIZIONE	IMPORTO AL 31/12/2003
Depositi cauzionali	2.625,00
Partecipazioni	34.143,00
Titoli di stato	27.186,00
Bilancio al 31/12/2003	pagina 9

(b)

Fig. 10.15. Additional lines and columns: (a) Input table; (b) output table.

the table, in particular in the first cell, there is an additional data w.r.t. the table in Figure 10.16(b). So this table is classified **found correctly**, while the cell highlighted in red will be classified as **merged external**.

Rows and Columns Not Recognized. When rows or columns that belong to the table are not extracted, the table is classified as *Partially Found Table*, cells belonging to the extracted table are normally classified, while the cells belonging to the rows or columns that have not been extracted are classified as *Not Recognized*. In Figures 10.17(b) the table will be classified as **partially found table** because the first has been not detected, cells within it are classified normally, whereas the cells of the first line are classified as **not recognized**.

DESCRIZIONE	S.DO INIZIALE	VARIAZIONI	S.DO FIN.
IMMOB. IMM.	107.590	784	108.374
IMMOB. MAT.	20.178	298	20.476
RIMANENZE	1.250	769	2.019
CREDITI	23.600	4.029	27.629
DISP. LIQUIDE	2.294	-2.221	73

(a)

E DEL PASSIVO DESCRIZIONE	S.DO INIZIALE	VARIAZIONI	S.DO FIN.
IMMOB. IMM.	107.590	784	108.374
IMMOB. MAT.	20.178	298	20.476
RIMANENZE	1.250	769	2.019
CREDITI	23.600	4.029	27.629
DISP. LIQUIDE	2.294	-2.221	73

(b)

Fig. 10.16. Wrongly data added to boundary cells (a) Input table; (b) output table.

Composizione	31/12/2003	variazione	31/12/2004
debiti verso banche	247.829	-85.641	162.188
debiti verso altri finanziatori	201.746	-27.649	174.097
altri debiti	69.545	-1.059	68.486
debiti verso fornitori	81.971	-23.200	58.771
debiti tributari	38.045	45.491	83.536
debiti verso personale e istituti di previdenza	202.568	68.281	270.849
Totale	841.704	-23.777	817.927

(a)

debiti verso banche	247.829	-85.641	162.188
debiti verso altri finanziatori	201.746	-27.649	174.097
altri debiti	69.545	-1.059	68.486
debiti verso fornitori	81.971	-23.200	58.771
debiti tributari	38.045	45.491	83.536
debiti verso personale e istituti di previdenza	202.568	68.281	270.849
Totale	841.704	-23.777	817.927

(b)

Fig. 10.17. Rows and Columns Not Recognized: (a) Input table; (b) output table.

Not Detected Table. If a table is not detected, it will be classified as *Not Found*, and the cells belonging to the table are also classified as *Not Found*.

Split Table Error. There are two cases of split of tables:

- When a table is split into two or more tables, the first table is classified as *Split Table*, while the other tables as *Extra Table*. An example is given in figure 10.18, where the input table is split in five tables, then first

is classified as `split table`, whereas the following four tables as `extra tables`.

- When the table is split but only across the header cells, whereas all data cells remain together. In this case the table containing all data is classified as *Data Cells Found*, and the additional tables as *Extra Table*. For instance, the input table 10.20 is split in the header (Figure 10.21) and in data table (Figure 10.22).

Sales	2,538	2,489	2,492	2,556	2,675
Var. %		-1.9	0.1	2.6	4.7
Value of production	2,075	2,009	1,968	2,033	2,070
Var. %		-3.2	-2.0	3.3	1.8
	0	0	0	0	0
Exports	586	600	608	657	708
Var. %		2.4	1.4	8.1	7.8
	0	0	0	0	0
Imports	830	954	1,030	1,194	1,245
Var. %		15.0	8.0	15.9	4.3
	0	0	0	0	0
Trade Balance	-244	-354	-422	-537	-537
	0	0	0	0	0
Domestic Consumption	4,450	4,549	4,662	4,948	5,140
Var. %		2.2	2.5	6.1	3.9
	0	0	0	0	0
Structural Indicators					
Exports/Sales	23.1	24.1	24.4	25.7	26.5
Normalized Trade balance	-17.2	-22.8	-25.8	-29.0	-27.5
Imports/Consumption at Production	29.8	33.5	35.3	38.6	38.8

Fig. 10.18. Input Table

Sales	2,538	2,489	2,492	2,556	2,675
Var. %		-1.9	0.1	2.6	4.7
Value of production	2,075	2,009	1,968	2,033	2,070
Var. %		-3.2	-2.0	3.3	1.8
Exports	586	600	608	657	708
Var. %		2.4	1.4	8.1	7.8
Imports	830	954	1,030	1,194	1,245
Var. %		15.0	8.0	15.9	4.3
Domestic	4,450	4,549	4,662	4,948	5,140
Var. %		2.2	2.5	6.1	3.9
Structural Indicators					
Exports/Sales	23.1	24.1	24.4	25.7	26.5
Normalized Trade balance	-17.2	-22.8	-25.8	-29.0	-27.5
Imports/Consumption at Production	29.8	33.5	35.3	38.6	38.8

Fig. 10.19. The table 10.18 split in five tables

Merge Table Error. When two or more separated tables are merged, the first table is classified as *Merged*, while the other tables are classified as *Not recognized*. All cells of the resulting tables are classified as normal. Figure 10.23(b) show an example of `merged table` in which two tables, shown in Figure 10.23(a), are merged into one. The part highlighted in green and in

Tipologie di contratto di cui comma 2.2 del Testo integrato		PC (centesimi di euro/kWh)
lettera a)	Utenza domestica in bassa tensione	6,25
lettera b)	Utenze in bassa tensione di illuminazione pubblica	5,47
lettera c)	Altre utenze in bassa tensione	6,80
lettera d)	Utenze in media tensione di illuminazione pubblica	5,19
lettera e)	Altre utenze in media tensione	6,57
lettera f)	Utenze in alta e altissima tensione	5,81

Fig. 10.20. An input table having a header not closely to data cells

Tipologie di contratto di cui comma 2.2 del Testo integrato	PC (centesimi di euro/kWh)

Fig. 10.21. Header of the table 10.20

lettera a)	Utenza domestica in bassa tensione	6,25
lettera b)	Utenze in bassa tensione di illuminazione pubblica	5,47
lettera c)	Altre utenze in bassa tensione	6,80
lettera d)	Utenze in media tensione di illuminazione pubblica	5,19
lettera e)	Altre utenze in media tensione	6,57
lettera f)	Utenze in alta e altissima tensione	5,81

Fig. 10.22. Data cells of the table 10.20

red corresponds to the first and second table respectively.

Nonexistent Table. Another mistake is extract tables that do not exist. These tables are classified as *Table Not Exist*, the cell within this table are classified as *Cells Not Exist*. Figure 10.24(b) represent a table recognized from Figure 10.24(a) where does not appear any table.

Classifications for Table Areas and Cells

In order to compute precision and recalls, the following mapping between TP, FP, TN, FN and the type of table and cells is defined in the following. In order to penalize partial correctly found tables and cells and maintain a value between $[0, 1]$, the numbers of: (i) cells in the dataset table (DD), (ii) cells that are extracted but that do not belong to the table (WE), (iii) cells that are extracted (EE) and (iv) cells that are not extracted (NE) are considered during the computation.

Direct Energy Content [TJ]	1994	1996	1998	2000	2004	2005	2006	2007	Change '94 - '07
Total Gross Electricity Production	144 708	192 879	147 998	129 776	145 583	130 468	164 199	140 964	-2.6%
Oil	9 547	20 808	17 906	15 964	5 881	4 933	5 811	4 616	-51.7%
- Orimulsion	-	14 495	12 890	13 467	7	-	-	-	*
Natural Gas	8 206	20 442	29 260	31 589	35 807	31 606	33 903	24 886	203%
Coal	119 844	142 795	85 151	60 022	67 232	55 665	88 439	71 631	-40.2%
Surplus Heat	-	123	136	139	40	-	-	-	*
Waste, non-renewable	463	610	702	994	1 163	1 459	1 472	1 416	206%
Renewable Energy	6 647	8 101	14 844	21 068	35 459	36 805	34 574	38 415	478%
Solar Energy	0	1	1	4	7	8	8	9	3 017%
Wind Power	4 093	4 417	10 152	15 268	23 699	23 810	21 989	25 823	531%
Hydro Power	117	69	98	109	95	81	84	101	-14.1%
Biomass	2 116	3 207	3 911	4 936	10 646	11 889	11 517	11 504	444%
- Straw	293	748	960	654	3 057	3 088	3 359	3 185	988%
- Wood	429	340	512	828	3 546	3 730	3 041	3 398	691%
- Waste, renewable	1 393	2 120	2 439	3 454	4 043	5 071	5 117	4 921	253%
Biogas	321	407	682	751	1 013	1 017	976	978	205%

Electricity from Renewables: Share of Domestic Electricity Supply ¹⁾	1994	1996	1998	2000	2004	2005	2006	2007	Change '90 - '07
Renewable Energy	5,2	5,9	11,2	15,9	26,2	27,2	24,8	27,9	435%
Solar Energy	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	*
Wind Power	3,2	3,2	7,7	11,6	17,5	17,6	15,8	18,8	484%
Hydro Power	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	-21%
Biomass	1,7	2,3	3,0	3,7	7,9	8,8	8,3	8,4	403%
- Straw	0,2	0,5	0,7	0,5	2,3	2,3	2,4	2,3	907%
- Wood	0,3	0,2	0,4	0,6	2,6	2,8	2,2	2,5	632%
- Waste, renewable	1,1	1,5	1,8	2,6	3,0	3,7	3,7	3,6	227%
Biogas	0,3	0,3	0,5	0,6	0,7	0,8	0,7	0,7	182%

¹⁾ Calculated in accordance with the EU Directive on electricity production from renewable energy sources, i.e. the renewable energy share has been calculated in relation to the overall gross electricity production including the imports of electricity. Calculated according to Energy Statistics guidelines, the share of electricity from renewable energy which increased with 20 % or more over 10 % year rate: renewable in 2007: 44 compared to 18 % in year rate over 10 year rate in 2006.

(a)

Direct Energy Content [TJ]	1994	1996	1998	2000	2004	2005	2006	2007	Change '94 - '07
Total Gross Electricity Production	144 708	192 879	147 998	129 776	145 583	130 468	164 199	140 964	-2.6%
Oil	9 547	20 808	17 906	15 964	5 881	4 933	5 811	4 616	-51.7%
- Orimulsion	-	14 495	12 890	13 467	7	-	-	-	*
Natural Gas	8 206	20 442	29 260	31 589	35 807	31 606	33 903	24 886	203%
Coal	119 844	142 795	85 151	60 022	67 232	55 665	88 439	71 631	-40.2%
Surplus Heat	-	123	136	139	40	-	-	-	*
Waste, non-renewable	463	610	702	994	1 163	1 459	1 472	1 416	206%
Renewable Energy	6 647	8 101	14 844	21 068	35 459	36 805	34 574	38 415	478%
Solar Energy	0	1	1	4	7	8	8	9	3 017%
Wind Power	4 093	4 417	10 152	15 268	23 699	23 810	21 989	25 823	531%
Hydro Power	117	69	98	109	95	81	84	101	-14.1%
Biomass	2 116	3 207	3 911	4 936	10 646	11 889	11 517	11 504	444%
- Straw	293	748	960	654	3 057	3 088	3 359	3 185	988%
- Wood	429	340	512	828	3 546	3 730	3 041	3 398	691%
- Waste, renewable	1 393	2 120	2 439	3 454	4 043	5 071	5 117	4 921	253%
Biogas	321	407	682	751	1 013	1 017	976	978	205%

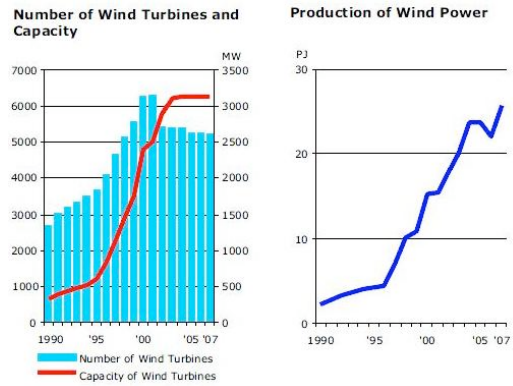
Electricity from Renewables: Share of Domestic Electricity Supply ¹⁾	1994	1996	1998	2000	2004	2005	2006	2007	Change '90 - '07
Renewable Energy	5,2	5,9	11,2	15,9	26,2	27,2	24,8	27,9	435%
Solar Energy	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0	0,0
Wind Power	3,2	3,2	7,7	11,6	17,5	17,6	15,8	18,8	484%
Hydro Power	0,1	0,1	0,1	0,1	0,1	0,1	0,1	0,1	-21%
Biomass	1,7	2,3	3,0	3,7	7,9	8,8	8,3	8,4	403%
- Straw	0,2	0,5	0,7	0,5	2,3	2,3	2,4	2,3	907%
- Wood	0,3	0,2	0,4	0,6	2,6	2,8	2,2	2,5	632%
- Waste, renewable	1,1	1,5	1,8	2,6	3,0	3,7	3,7	3,6	227%
Biogas	0,3	0,3	0,5	0,6	0,7	0,8	0,7	0,7	182%

(b)

Fig. 10.23. Merged Table Error: (a) Input tables; (b) output table.

Found correctly	TP
Data cells found	TP
Split table	TP
Extra table	FP
Merged into surroundings	TP
Merged	TP
Not recognized	FN
Not found	FN
Partially table	$TP = \frac{DD-NE}{DD}$
Table not exist	FP

Table 10.3. Classifications for table areas



(a)

Number of Wind Turbines and Capacity M				Production of Wind Power PJ			
7000	3500			30			
6000	3000						
5000	2500			20			
4000	2000						
3000	1500			10			
2000	1000						
1000	500			0			
0	0						
1990	'95	'00	'05 '07	1990	'95	'00	'05 '07
Number of Wind Turbines							
Capacity of Wind Turbines							

(b)

Fig. 10.24. (a) Input PDF document; (b) “nonexistent” recognized table.

Found correctly data	TP
Found correctly blank	TP
Split full	TP
Split data	$TP = \frac{DD}{EE}$
Split blank	TP
Void data	FP
Void blank	FP
Extra data	FP
Extra blank	FP
Merged	$TP = \frac{EE}{DD}$
Cells not found	FN
Cells not exist	FP
Merged external	$TP = \frac{DD}{WE+DD}$

Table 10.4. Classifications for cells

10.2 SILA: Extracting Data Records from Deep Web Sites

In this section is presented the Spatial Instance Learning Approach (SILA), a wrapper induction method that exploits the spatial arrangement of Deep Web page produced by layout engines of web browsers. SILA is founded on: (i) a spatial similarity function that takes into account visual cues available after document rendering that help human readers to make sense of page contents independently from the internal structure of the web page; (ii) the definition of a very efficient and effective wrapper induction algorithm based only on hierarchical clustering that make no use of the internal structure of web pages. SILA analyzes the SDOM of Deep Web pages, introduced in Section 9.3, and learns wrappers that allow for extracting information from Web pages and storing them in structured form. Main innovations proposed by SILA are: (i) the direct exploitation of the structure of layouted page without using intermediate algorithms that interpret the spatial representation of the page; (ii) the ability to identify all data records of a Deep Web page independently from spatial patterns adopted for arranging the contents of deep web pages. This way SILA allow for recognizing data records in a wide variety of deep web sites. In the following are given motivations that draw the definition of the SILA wrapper induction method and then are shown: (i) the novel spatial similarity measure between nodes of a web page represented by the spatial document model; (ii) the data regions and data records identification algorithms. Finally, experimental results obtained from a dataset of 100 Deep Web pages are given and discussed.

Motivations

Web designers plan web pages contents in order to provide visual patterns that help human readers to make sense of document contents. This aspect is particularly evident in *Deep Web* pages [122], where designers always arrange data records and data items with visual regularity to meet the reading habits of humans. Figure 9.6 shows a characteristic Deep Web page in which are contained a set of repetitive records. All data records are generated through a query on a database in the back-end of the web site. Data items of the data records (i.e. attributes of the corresponding tuples in the database) are arranged by using the same layout.

Existing wrapper induction approaches (e.g. [37, 198]) have exploited regularities in the underlying document structures, which led to such similar layout, to translate such information into relational or logical structures. However, surveying a large number of real (deep) web pages, we have observed that the document structure of current Web pages has become more complicated than ever implying a large conceptual gap between document structure and layout structure. In particular, we discovered that data records can be

internally represented by different HTML tree structures. More in detail we discovered that data records are organized by the following four arrangement:

- *standard* when a data record is identified by a parent node in SDOM that contains all other nodes representing record items. In this case data items are all child of a single SDOM node.
- *flat* when in the SDOM data records are spread in multiple nodes that contain only portions of the data records (e.g. a single data item). In this case there is no a parent nodes that groups all data items belonging to a single record. Wrapper induction for this kind of data records arrangement is very difficult.
- *nested* when data records are organized in groups each having a parent node. In this case the are nodes in the SDOM that contain multiple data records.
- *flat nested* when nodes of the SDOM contain other nodes that, in turn, contain pieces of final records. This is the most difficult case, because there are no nodes that contain a complete data records, but the are nodes that contain multiple portion of many data records.

Such kind of arrangements are made very difficult for existing wrapper induction approaches (e.g. [75, 198]) to infer the regularity of the structure of Deep Web pages [31] by only analyzing the intricate tag structure. Hence, the effectiveness of existing wrapper induction methods suffers from the requirement to analyze HTML documents with increasing structural complexity. At the contrary human readers can exploit visual patterns that they recognize on the screen. In particular a human reader can relate information on the page in Fig. 9.6 by considering the spatial arrangement of layouted elements. He can interpret the spatial proximity of images and nearby strings as a corresponding aggregation of information, namely as the complete record describing the details of a product and one of its photos. So the SILA approach exploits visual patterns, instead of DOM structures, for inferring web wrappers.

Spatial Similarity

The main idea which the spatial similarity is founded on is that two nodes are spatially similar if they contain a set of syntactic similar leaf nodes arranged in a similar way. Two nodes are syntactically similar if they have the same name (function *name()*) and a shared set of attributes. In order to formally define spatial similarity we define the spatial context as follows.

Definition 10.11. *The spatial_context of a node is the a set of 3-tuples of the form $\langle u, w, r \rangle$ where u , and w are leaf nodes in V_v and r is the rectangle algebra relation between u and w .*

The function *spatialContext* : $V_v \rightarrow 2^{V_v \times V_v \times RA}$ computes the spatial context of n . The *spatial similarity* between two SDOM nodes n_1 and n_2 is computed by the Algorithm 2, where the function *synSimilarity* 3 computes the syntactic similarity between two nodes.

Algorithm 2: spatialSimilarity

Input : Two nodes n_1 , and n_2 and threshold λ
Output: Spatial similarity between n_1 , and n_2 in $[0, 1]$

```

1  $l_1 := \text{getLeafs}(n_1)$ ;
2  $l_2 := \text{getLeafs}(n_2)$ ;
3 if  $|l_1| = 0 \vee |l_2| = 0$  then
4 | return 0;
5 end
6 if  $|l_1| = 1 \wedge |l_2| = 1$  then
7 | return  $\text{synSimilarity}(\text{getFirst}(l_1), \text{getFirst}(l_2))$ ;
8 end
9  $e' := \text{spatialContext}(l_1)$ ;
10  $e'' := \text{spatialContext}(l_2)$ ;
11 if  $|e'| > |e''|$  then
12 | swap( $e', e''$ );
13 end
14  $\text{maxCardinality} := |e''|$ ;
15  $s := \emptyset$ ;
16 foreach  $(src', dest', ra') \in e'$  do
17 | foreach  $(src'', dest'', ra'') \in e''$  do
18 | | if  $\text{synSimilarity}(src', src'') \geq \lambda \wedge$ 
19 | |    $\text{synSimilarity}(dest', dest'') \geq \lambda \wedge$ 
20 | |    $ra' = ra''$  then
21 | | |  $s := s \cup \{src'\} \cup \{dest'\}$ ;
22 | | |  $e'' := e'' - \{src'', dest'', ra''\}$ ;
23 | | | break;
24 | | end
25 | end
26 return  $\frac{|SET|}{\text{maxCardinality}}$ ;

```

Algorithm 3: synSimilarity

Input : Two nodes n_1, n_2
Output: The syntactic similarity between n_1, n_2

```

1 if  $\text{name}(n_1) \neq \text{name}(n_2)$  then return 0;
2  $a_1 := \text{attributes}(n_1)$ ;
3  $a_2 := \text{attributes}(n_2)$ ;
4  $\text{maxCard} := \max(|a_1|, |a_2|)$ ;
5 if  $\text{maxCard} = 0$  then return 1;
6  $SET := a_1 \cap a_2$ ;
7 return  $\frac{|SET|}{\text{maxCard}}$ ;

```

10.2.1 The SILA Algorithm

The Algorithm 4 implements the SILA wrapper induction method. The algorithm takes in input a the SDOM representation of a Deep Web page and returns a set of data records with aligned data items. In instruction 1, the algorithm evaluates all possible data regions in a Web page by the procedure *getDataRegions*. The instruction 2 computes the data region having the greatest area. By instruction 3 data regions similar to the greatest data region are computed by means of the procedure *findSimilarRegion*. instruction 4 gets

data record contained in all data regions identified in the Web page. Finally, instruction 5 returns aligned data items present in all recognized data records.

Algorithm 4: SILA

Input : A *SDOM*
Output: Data Records with Aligned Data Items *RECs*

- 1 *REGs* := `getDataRegions(getRoot(DOC), ϵ)`;
- 2 *REG* := `maximalDataRegion(REGs, μ)`;
- 3 *REGs'* := `findSimilarRegions(REG, REGs)`;
- 4 *RECs* := `getDataRecords(REGs')`;
- 5 **return** `getDataItems(RECs)`;

The Algorithm 5 (*getDataRegions*) computes a list of all possible data regions. In instruction 1 the algorithm gets all children node of a node n . Instruction 2 computes a data region by means of the *computeDR* procedure that takes in input a parent node n and all its children. Instructions 4, 6 perform a depth-first search along the containment tree in the *SDOM* in order to find all possible data regions.

Algorithm 5: getDataRegions

Input : A node n and a similarity threshold ϵ
Output: All the possible Data Region *REGs* in n

- 1 *children* := `getChildrenOf(n)`;
- 2 *REG* := `computeDR(children, n, ϵ)`;
- 3 *REGs* := {*REG*};
- 4 **foreach** $n' \in \text{children}$ **do**
- 5 | *REGs* := *REGs* \cup `getDataRegions(n')`
- 6 **end**
- 7 **return** *REGs*;

The Algorithm 6 (*computeDR*) evaluate if a node n contains a data region and return it if it exists. The Algorithm takes in input a node, the set of its children nodes and a spatial similarity threshold. The threshold has been experimentally tested in thousand Depp Web pages. In instruction 2 the algorithm computes in *SN*, groups of similar nodes (procedure *clusterize*) by using the single linkage clustering strategy and the spatial similarity measure defined in Algorithm 2. By instruction 4 the algorithm heuristically filters clusters that contain a number of nodes 30% less than the average number of nodes in all clusters. In instruction 5 the algorithm rearranges (procedure 7) clusters in order to build a new group of *dummy nodes* each representing a data record. Dummy nodes are introduced in order to represent final records or set of records by a single parent node when the spatial arrangement of nodes is nested or flat nested. In the standard spatial arrangement of data records dummy nodes coincide with the parent node that contain all data

items of a record. By instructions 8-22 the algorithm performs a search for discovering nested, flat and flat nested record structures.

Algorithm 6: computeDR

Input : A node n , the set S of its children nodes, and a similarity threshold ϵ
Output: A Data Region REG

```

1   $REG := \emptyset$ ;
2   $SN := \text{clusterize}(S, \epsilon)$ ;
3  if  $SN \neq \emptyset$  then
4     $SN := \text{filter}(SN)$ ;
5     $dummyNodes := \text{regroup}(REG)$ ;
6     $dummyNodesSize := |dummyNodes|$ ;
7     $done := \text{false}$ ;
8    while  $\neg done$  do
9       $list := \emptyset$ ;
10     foreach  $node \in dummyNodes$  do
11        $list := list \cup \text{getGrandChildrenOf}(node)$ ;
12     end
13      $SN' := \text{clusterize}(list, \epsilon)$ ;
14      $tmpDummyNodes := \text{regroup}(SN')$ ;
15      $SN'' := \text{clusterize}(tmpDummyNodes, \epsilon)$ ;
16     if  $|SN''| = 1 \wedge (|tmpDummyNodes| > dummyNodesSize)$  then
17        $dummyNodesSize := |tmpDummyNodes|$ ;
18        $dummyNodes := tmpDummyNodes$ ;
19     else
20        $done := \text{true}$ ;
21     end
22   end
23    $REG.REC := dummyNodes$ ;
24    $REG.NODE := n$ ;
25 end
26 return  $REG$ ;

```

The Algorithm 7 (*regroup*) computes dummy nodes by regrouping nodes belonging to different clusters of similar nodes. In instruction 1-3, for each cluster c , a spatial ordering among nodes is built on the base of node positions on the web page. In particular, nodes are sort from top to bottom and from left to right. In instruction 4 the algorithm spatially orders clusters in the set C on the base of the spatial position of the top-left nodes. More in detail, clusters are sort from top to bottom and from left to right (considering only the top left nodes). Instruction 5 computes the seed cluster, that is, the cluster having the greatest number of nodes. Instructions 8-13 create a dummy nodes, assign to the dummy nodes all remaining nodes as children and computes the MBR of the dummy nodes by considering all MBRs of children (instruction 12). In instructions 14-18 for each dummy node, closest nodes are found and assigned to it (closeness is computed by considering the distance between the center of the MBR of the dummy node and the center of the MBR of the analyzed nodes). This way the dummy node will be parent of a set of close nodes that constitute the data record. The MBR of a dummy node is recomputed by considering MBRs of all children nodes.

Algorithm 7: regroup

Input : A set of cluster C
Output: The set of cluster C containing regrouped Data Records

```

1 foreach  $c \in C$  do
2   | sortIntraCluster( $c$ );
3 end
4 sortInterCluster( $C$ );
5  $seed := \text{findBiggestCluster}(C)$ ;
6  $C := C - \text{biggestCluster}$ ;
7  $V := \emptyset$ ;
8 foreach  $n \in seed$  do
9   |  $dummyNode := \text{createPosNode}()$ ;
10  |  $\text{setParent}(n, dummyNode)$ ;
11  |  $V := V \cup \{dummyNode\}$ ;
12  |  $\text{reAllocateMBR}(dummyNode)$ ;
13 end
14 foreach  $c \in C$  do
15   | foreach  $n \in c$  do
16     |  $findClosestNode(cn, V)$ ;
        /* la seguente unisce due nodi in termini di area
        e li assegna ad un unico nodo padre fittizio.*/
16     |  $\text{setParent}(n, cn)$ ;
16     |  $\text{reAllocateMBR}(cn)$ ;
17   | end
18 end
19  $C := \{V\}$ ;
20 return  $C$ ;
```

The Algorithm 8 (*maximalDataRegion*) computes the region that has the greatest area (among those computed by *getDataRegion*).

Algorithm 8: maximalDataRegion

Input : A List of Data Regions $REGs$ and a threshold number of records μ
Output: A Data Region REG with the biggest area and at least μ records inside

```

1  $sumax := 0$ ;
2  $BESTreg := null$ ;
3 foreach  $REG \in REGs$  do
4   |  $CL := \text{getFirst}(REG)$ ;
5   | if  $|CL| \geq \mu$  then
6     |  $sum := 0$ ;
7     | foreach  $NODE \in CL$  do
8       |  $sum := sum + \text{area}(mbr(NODE))$ ;
9     | end
10    | if  $sum > sumax$  then
11      |  $sumax := sum$ ;
12      |  $BESTreg := REG$ ;
13    | end
14  | end
15 end
16 return  $BESTreg$ ;
```

Algorithm 9: getDataRecords

Input : A set of Data Region REGIONS
Output: A set of records RECs containing data items. Every record in RECs is an array of nodes

```

1 RECs := ∅;
2 foreach REG ∈ REGIONS do
3   foreach ROOT ∈ REG.REC do
4     create an array V;
5     extract images, text nodes and put them into V;
6     RECs := RECs ∪ V;
7   end
8 end
9 return C;

```

Algorithm 10: getDataItems

Input : A set of regrouped records RECs where every record is an array of items, every item is a positional node
Output: Aligned records in a n*m matrix of positional nodes, n is the number of records retrieved in the web page, every row contains almost m items belonging to a record

```

1 //find the record R with the highest number of item inside;
2 RECs := RECs - R;
3 for i ← 1 to |R| do
4   M1,i := Ri;
5 end
6 for i ← 1 to |RECs| do
7   R' := RECsi;
8   for j ← 1 to |R'| do
9     N1 := R'j;
10    for k ← 1 to |R| do
11      N2 := Rk;
12      SS := synSimilarity(N1, N2);
13      ED := 1 - editDist(valueOf(N1), valueOf(N2));
14      TEMPj,k :=  $\frac{SS+ED}{2}$ ;
15    end
16  end
17  for j ← 1 to |R'| do
18    (maxj, maxk) := nextMax(TEMP);
19    Mi+1, maxk := R'maxj;
20  end
21 end
22 return M;

```

10.2.2 Experiments

Evaluation Measures

Like for the experiments performed for PDF-TREX tool (see Section 10.1.2), to measure the goodness of data extracted have been used Precision (P), Recall (R), and F-Measures (F). A dataset of 100 heterogenous HTML documents is used. The structure of the data records may be flat (Figure 10.25), nested (Figure 10.26) or standard (Figure 10.27).

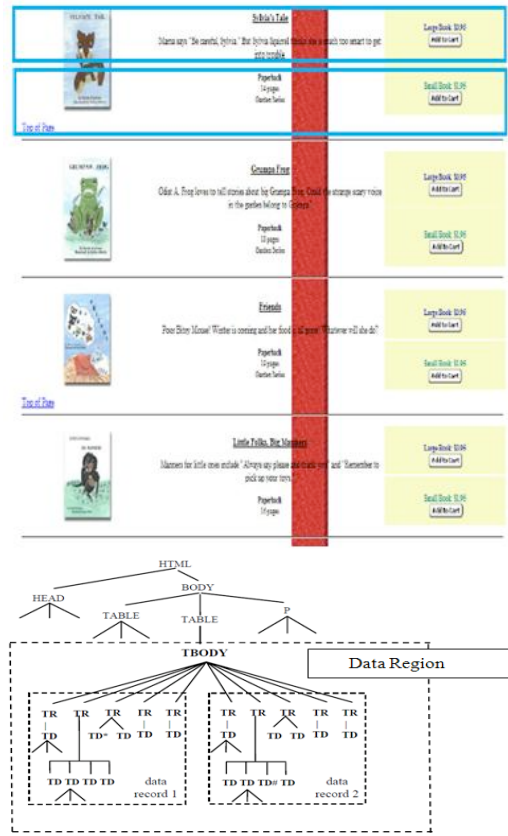


Fig. 10.25. Flat Structure

Data region structure may be vertical or horizontal. Moreover, the document may have a single region or a multi region (Figure 10.28).

Experimental Results

The Table 10.5 reports the precision, recall and F-measure calculated for the SILA system and compares its results with the MDR [114] and ViNTs [201] systems, which are available for download and for browser queries respectively. It is noteworthy that MDR and ViNTs allow for performing data record extraction, but they do not allow the data item extraction. However, both system implemented a more complete version that allow both steps, but that are not available on internet.

For the experiments, a dataset

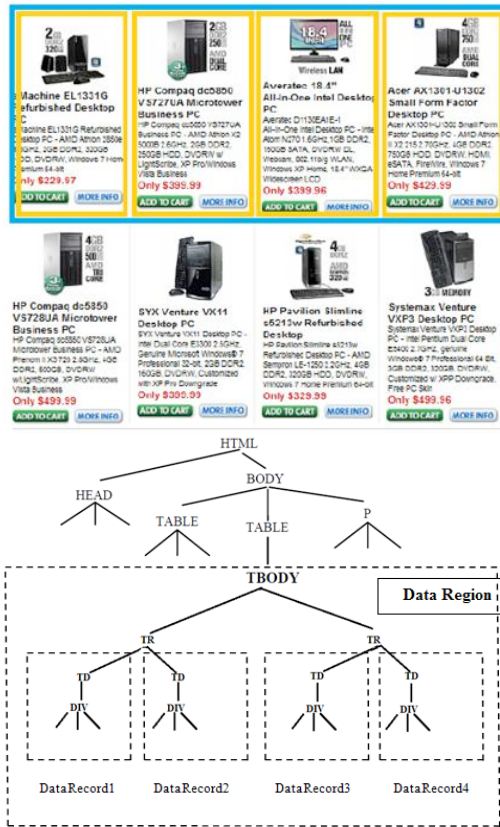


Fig. 10.26. Nested Structure

0	Tables			Cells		
	P	R	F	P	R	F
SILA	96.01%	94.33%	95.16%	93.62%	99.01%	96.24%
MDR	24.26%	42.85%	30.98%	—	—	—
ViNTs	51.52%	47.46%	49.41%	—	—	—

Table 10.5. Comparison of precision and recall results for both table areas and cells



Fig. 10.27. Standard Structure

Evaluation Method

Data Record Errors

A record correctly recognized is classified as *Found Correctly Record*. At this stage, we do not take care about the correctness of the data items, but we only consider the “area” of the record.

Not Recognized Record. This error occur when a record is not recognized at all.

Not Correctly Found Record. This error occur when:

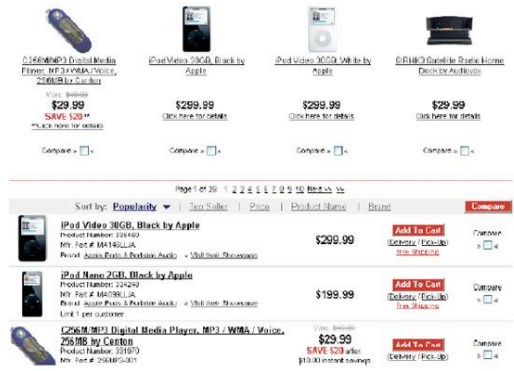


Fig. 10.28. Html fragment with a vertical region and a horizontal region

- are extracted not real records. For instance, the Figure 10.29(b) shows false positive records extracted from the input Web page shown in Figure 10.29(a);
- separated records are merged. For instance, records shown in Figure 10.30(a) are merged as shown in figure 10.30(b);
- two or more records are split.

Data Item Errors

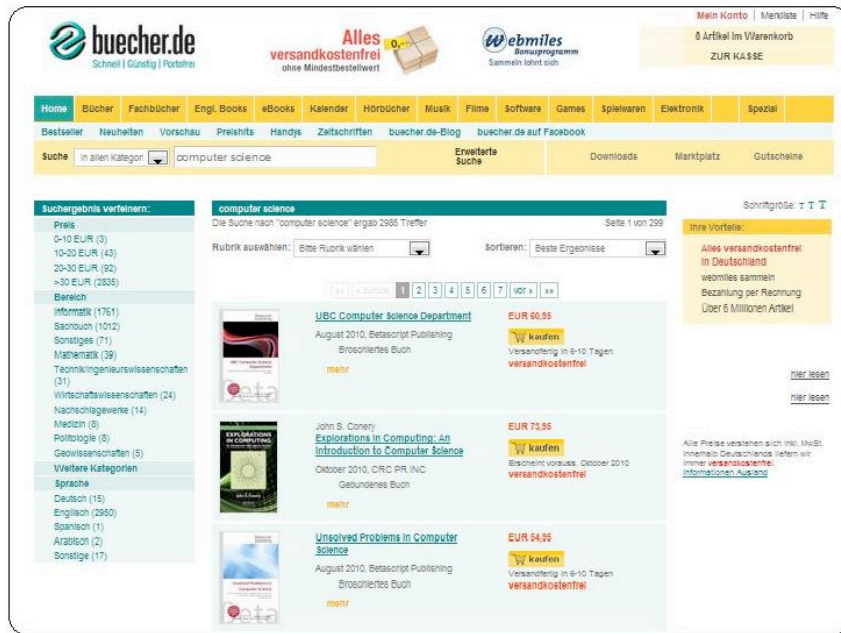
An item correctly found and aligned with the other items of the same type, is classified as *Found Correctly Item*. For instance, Figures 10.31(a) and 10.31(b) represent the input Web page and the output records with aligned items represented in a table, respectively.

Not Recognized Item. This error occur when a record is not detected at all.

Not Correctly Found Item. An Item is not correctly found if it is not correctly aligned to the other items of the same type. For instance, the input page shown in Figure 10.32(a) contains three records. As shown in 10.32(b) all three records are correctly extracted, but two items highlighted in blue are not aligned properly (because the second position should contain the name of the model of the bike, and not information about the negotiations that should be at the fourteenth place). So, they are classified as **not correctly found items**, while the other items are classified as **correctly found items**.

Classifications for Records and Items

In Figure 10.6 is defined the mapping between the classification adopted and the parameters that assess the quality of the systems.



(a)

Data Record 1

Rubrik auswählen:

Sortieren:

Data Record 2

Suche

[Erweiterte](#)

[Suche](#)

Data Record 3

[AGB](#) | [DATENSCHUTZ](#) | [WIDERRUFSBELEHRUNG](#) | [IMPRESSUM](#) | [KONTAKT](#) | [SITEMAP](#)
 KATEGORIEN: [START](#) | [BuCHER](#) | [ENGL. BuCHER](#) | [EBOOKS](#) | [KALENDER](#) | [HoRBuCHER](#) | [MUSIK](#) |
 SITEMAP: [1](#) | [2](#) | [3](#) | [4](#) | [5](#) | [6](#) | [7](#) | [8](#) | [9](#) | [10](#)

(b)

Fig. 10.29. Extraction of wrong records: (a) input Web page; (b) extracted records

The screenshot shows the SMART BARGAINS.COM website interface. At the top, there is a navigation bar with categories like Bed & Bath, Home & Kitchen, Women's, Men's, Jewelry & Watch, and Shoe. A search bar contains the text 'Search for your bargains'. To the right, there is a promotional banner for 'The Big Bad Bedding Event' and a sign-up form for hot deals. Below the navigation, a sidebar lists various bargain categories. The main content area displays 'Search Results' for the query 'jewel', showing 29 results. Three items are visible: a Belmont Jewel Top Comforter Set (On Sale), a Waterford Ballet Ribbon Jewel Box Set, and an Elie Tahari Simona Blouse with Jeweled Trim (Clearance). Each item includes a product image, title, price comparison, and availability information.

(a)

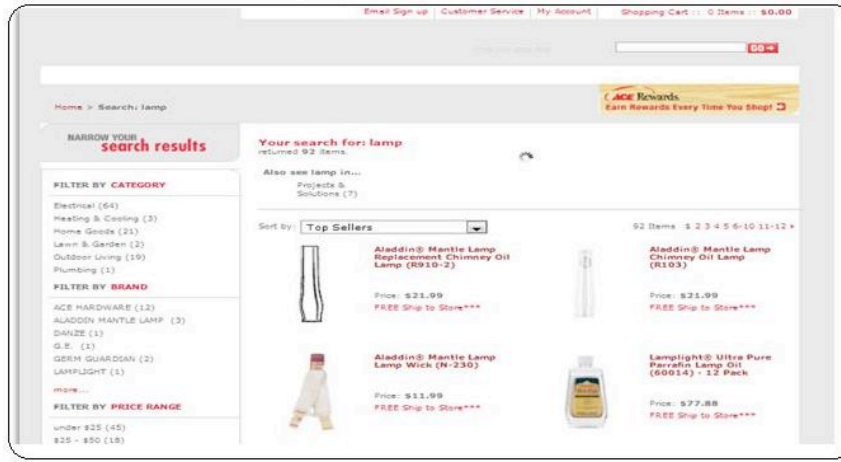
Data Record 1

This block displays the extracted data records for the three items from the search results. Each record is structured as follows:

- Item 1:** Belmont Jewel Top Comforter Set. Status: On Sale. Compare at: \$199.99. Sale: \$89.98. You Save: \$110.01 (55%). Almost Gone: Only 9 Left. Customer Reviews: 2 reviews.
- Item 2:** Waterford Ballet Ribbon Jewel Box Set. Compare at: \$66.00. Our Price: \$49.99. You Save: \$20.02 (23%). Almost Gone: Only 10 Left.
- Item 3:** Elie Tahari Simona Blouse with Jeweled Trim. Status: Clearance. Compare at: \$298.00. Clearance: \$70.97. You Save: \$227.03 (76%). Almost Gone: Only 2 Left.

(b)

Fig. 10.30. Merging of records: (a) input Web page; (b) extracted records



(a)

DATARECORD1		Aladdin® Mantle Lamp Replacement Chimney Oil Lamp (R910-2)	Price: \$21.99	FREE Ship to Store***
DATARECORD2		Aladdin® Mantle Lamp Chimney Oil Lamp (R103)	Price: \$21.99	FREE Ship to Store***
DATARECORD3		Aladdin® Mantle Lamp Lamp Wick (N-230)	Price: \$11.99	FREE Ship to Store***
DATARECORD4		Lamplight® Ultra Pure Paraffin Lamp Oil (60014) - 12 Pack	Price: \$77.88	FREE Ship to Store***

(b)

Fig. 10.31. Correctly extracted records and items: (a) input Web page; (b) extracted records and items

Data Record	
Correctly Found Record	<i>TP</i>
Not Correctly Found Record	<i>FP</i>
Not Recognized Record	<i>FN</i>
Data Item	
Correctly Found Item	<i>TP</i>
Not Correctly Found Item	<i>FP</i>
Not Recognized Item	<i>FN</i>

Table 10.6. Classifications for Records and Items

The screenshot shows the Motomercato website interface. At the top left is the logo with the URL www.motomercato.it. On the right, there are buttons for 'RICERCA' and 'PUBBLICITÀ', and a login section with the text 'Accedi a Motomercato.it', 'Email', 'Password', and 'accedi'. Below the logo is a sidebar with search filters: 'moto nuove', 'moto usate', 'moto aziendali', 'moto km 0', 'moto rivenditori', 'moto privati', 'concessionarie', and 'Le Prove'. The main content area shows 'Oggetti trovati: 227' with 'INDIETRO' and 'RICERCA' buttons. A dropdown menu 'Ordina per' is set to 'Seleziona'. Three motorcycle listings are displayed in a grid:

- HONDA VFR**: Km: M/A: /2010, CC: , Colore: Provincia: LU, Nuovo, Prezzo: Trattativa riservata » DETTAGLI
- APRILIA DORSODURO 750 FACTORY**: Km: M/A: /2010, CC: 750, Colore: Provincia: LU, Nuovo, Prezzo: Trattativa riservata » DETTAGLI
- HONDA CB1000R**: Km: M/A: /2010, CC: 1000, Colore: Provincia: SI, Nuovo, Prezzo: Trattativa riservata » DETTAGLI

(a)

DATA RECORD6		HONDA	VFR	Km:	MA:	/2010	CC:		Colore:	Provincia:	LU	Nuovo	Prezzo:	Trattativa riservata	» DETTAGLI
DATA RECORD7		APRILIA	DORSODURO 750 FACTORY	Km:	MA:	/2010	CC: 750		Colore:	Provincia:	LU	Nuovo	Prezzo:	Trattativa riservata	» DETTAGLI
DATA RECORD8		HONDA	Trattativa riservata	Km:	MA:	/2010	CC: 1000		Colore:	Provincia:	SI	Nuovo	Prezzo:	CB1000R	» DETTAGLI

(b)

Fig. 10.32. Extracted records with items not aligned: (a) input Web page; (b) extracted records and items

10.3 Discussion

In this section have been presented: (i) the PDF-TREX approach that allows for understanding PDF documents structure and recognizing and extracting tables from them, (ii) the SILA wrapper induction approach that allow for learning wrapper by using only the spatial arrangement of content items of Deep Web pages.

The PDF-TREX approach uses the 2-Dimensional flat document representation of PODs. The approach has been designed for recognizing a wide variety of table layouts and does not use any graphical or linguistic document feature. It aligns and groups content items in order to identify tabular arrangement of information. The approach is able to detect multi-line row headers and column headers spanning on multi row/column. The output of the approach is a set of 2-dimensional table cells that form a grid. Table cells can be seen as initial bricks well suited for understanding table layouts and contents by using spatial reasoning. PDF-TREX approach features contribute to improve document annotation and information extraction from PDF documents. The evaluation of system performances on a dataset composed of 100 documents and 164 tables shows that the system can be considered the state of the art in recognizing and extracting tables from PDF documents. In the future this approach will be also applied to the recognition and extraction of tables from Web pages.

The SILA approach uses the SDOM representation of PODs. It allows for inferring wrappers for Deep Web pages by analyzing only the spatial arrangement of content items produced by the layout engine of a Web browser. So SILA do not requires external algorithms that produce specific representation of Web pages starting from the HTML DOM as happens, for instance, in the VIDE approach [116]. Furthermore, the SILA approach allows for overcoming limitations of wrapper induction approaches based on the tag structures. In fact, tag structures of current web pages is very involved and undergoing, difficult to process for wrapper induction methods like MDR [114] and ViNTS [201] as shown in experiments. Experiments also show that SILA in very effective with high level of precision and recall on a wide variety of real Deep Web pages.

Querying PODs

Querying data from presentation formats like HTML or PDF, for purposes such as information extraction, requires a versatile machinery to abstract from presentation structure into conceptual relationships. Germane to such abstraction is the transformation into structural languages such as XML. Indeed quite often XML languages are used as presentation formats, e.g. HTML 5. Based on these structural languages, one may use well founded and known query formalisms such as XPath and XQuery (see Section 2.2) in order to turn presentation structure into meaningful relational or logical representations. Unfortunately, as described in Section 2.1, the tree-based structures of XML are often not convenient and sometimes even not expressive enough in order to represent the presentation layout and its corresponding conceptual relationships that is indicated to the reader of such documents by its presentation.

Chapter 7 describes approaches and frameworks proposed in literature that are aimed at manipulating Web pages by leveraging the visual arrangement of page contents, and representing and querying multimedia and presentation databases. However, such approaches and frameworks provide limited capabilities in navigating and querying Web documents for information extraction purposes. Therefore, we have proposed *Spatial XPath* (SXPath), an extension of XPath 1.0 that allows for inclusion of spatial navigation primitives into the language resulting in conceptually simpler queries on Web documents. SXPath enables to navigate the innovative Spatial Document Object Model (SDOM), presented in Section 9.3, which constitutes a mixed spatial and structural hierarchical model (DOM). The SXPath language is based on a combination of a spatial algebra with formal descriptions of XPath navigation, and maintains polynomial time combined complexity. Practical experiments have demonstrated the usability of SXPath. This language will be first presented at VLDB 2011 [145].

In order to recognize a wide variety of content structures (e.g. repetitive records, tables, news, infoboxes, profiles, etc) by exploiting their spatial arrangement, SXPath may be used as building block of more expressive

languages, such as XQuery. Whereas XPath and XQuery 2.2 are well known Web Query languages, grammars 2.3 represent the more formal syntactic descriptions of languages, indeed, they are the starting point for descriptions of the syntax of natural human languages. In PODs, a lot of information is represented in natural language, therefore we investigated as combine the spatial construct with the power of the grammars. Thus, a *Spatial Grammar* (SG) is defined and a query on a document is constituted by a set of *spatial production rules*. Work related to spatial grammars has been published at ICTAI 2010, and ICAART 2011 [135, 136] and it is related to [142].

11.1 SXPath: Spatial Querying of PODs

As described in the Section 2.1, presentation-oriented documents like PDF and HTML are human oriented and the spatial arrangements help humans to understand the semantic of contained information. This aspect is particularly evident in *Deep Web* pages [122], where designers always arrange data records and data items with visual regularity to meet the reading habits of humans or in professional documents like invoices that contains a standard content arrangement and tables. In Section 2.1 we have shown examples deep web pages, where records have the similar structure and are clearly recognizable.

The SXPath language allows for navigating SDOM, thus it allows for querying the internal structure of pages like XPath 1.0, as well as for exploiting the spatial layout of DOM nodes. Therefore, SXPath can be used not only for classical Web pages after their rendering, but also other types of presentation oriented documents. In the following examples we give an intuition about novel features and capabilities of the SXPath language.

In the rendered page each DOM node is visualized in a rectangle having sides parallel to the axes of the Cartesian plane. Fig. 11.1 depicts rectangles created by a layout engine for the page in Fig. 2.2. Some rectangles are annotated by fragments of XPath 1.0 location paths that characterize related DOM nodes.

The Web page in Fig. 2.2 allows us to show how to extract data about music bands by a corresponding conventional query using existing XQuery 1.0 and XPath 1.0 (see Section 2.2) on the DOM partially depicted in Fig. 2.3.

Example 11.1. XQuery 1.0 and XPath 1.0

```

for $li in document ("last-fm.htm")
(1.1) //div[@id='content']//ul/li  return
      <music-band>
(1.2) <name> {$li/a/strong/text()} </name>
      <photo>
(1.3)      {$li/a/span/span/img}
      </photo>

```

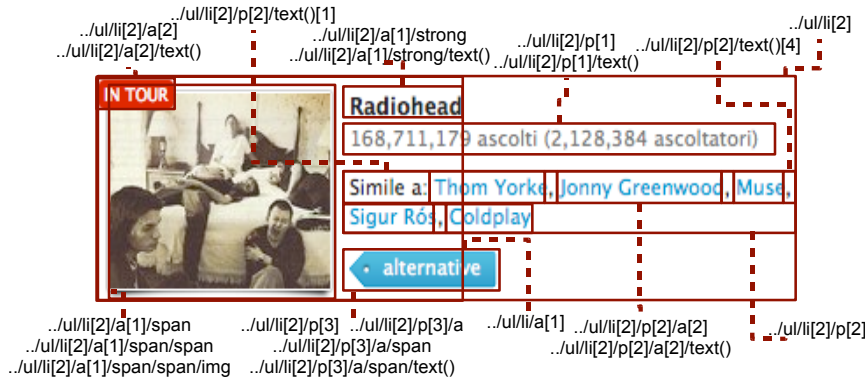


Fig. 11.1. Rectangles that Bound Visualized DOM Nodes

```

(1.4) <number-of-listening-and-listeners>
      {$li/p[1]/text()}
      </number-of-listening-and-listeners>
<similar-bands>
(1.5)   {$li/p[2]//text()}
</similar-bands>
<genre>
(1.6)   {$li/p[3]/a/span/text()}
</genre>
</music-band>

```

For writing the query in Example 11.1 above, a human user must know the intricate DOM structure of the input Web page (a sketch of which is given in Fig. 2.3). The intricate DOM structure makes it difficult for the user to pose this query. The location path (1.1) is the shortest relative path found by performing many attempts. In fact, shorter location paths like `//li` or `//div//ul/li` (that does not use attributes) make the query unsound.

Examples in the following show how the spatial arrangement can be exploited for navigating and querying content items on the base of the visual interpretation given by a human reader. Examples give an intuitive explanation of XPath capabilities. Formal syntax and semantics of XPath are more rigorously presented and discussed in Section 11.1.1.

Example 11.2. XQuery 1.0 and XPath

In contrast of the Example 11.1 above, the following query exploits XPath that allows for extracting details of music bands by exploiting only the DOM nodes of type `img` and `text`, and their spatial relationships.

A human reader can relate information on the page in Fig. 2.2 by considering the spatial arrangement of laid out content elements. He can interpret the spatial proximity of images and nearby strings as a corresponding aggregation of information, namely as the complete record describing the details of

a music band profile and one of its photos. *A music band profile is: the music band photo that has at east its descriptive information. That is its name, a number of hits, a list of similar music artists and a music genre, which are on top of each other.*

```

for $img in document ("last-fm.htm")
(2.1) /CD::img[N|S::img] return
    <music-band>
(2.2) <name> {$img/E::text[W,1][N,1]} </name>
    <photo>
(2.3)     {$img}
    </photo>
    <number-of-listening-and-listeners>
(2.4)     {$img/E::text[E,1][N,2]}
    </number-of-listening-and-listeners>
    <similar-bands>
(2.5)     {$img/E::*[W,1][N,3][max]/CD::text}
    </similar-bands>
    <genre>
(2.6)     {$img/E::text[E,1][S,1]}
    </genre>
</music-band>

```

The *spatial location path* (2.1) returns images that vertically listed in the page. These are exactly the photos of music bands. Such kind of visual patterns (i.e. alignment in a given direction) are very frequent in Deep Web pages, but often hard to recognize in the DOM structure.

The spatial location path `$img/E::text` in (2.2) returns all nodes labeled by `text` that lie on east (spatial axis E) of the context node represented by the variable `$img` (photos of music bands). Among these nodes the predicates `[W,1]` and `[N,1]` select the node that is the first from west and from north, i.e. the name of the bands (e.g. `Coldplay` and `Radiohead`).

The spatial location path (2.5) selects the nodes that are first from west, third from north, and are not contained in other nodes (predicate `[max]`) among all nodes on east of each photo of music bands (see the node `..ul/li[2]/p[2]` in Fig. 11.1). So, all similar bands are selected.

Example 11.2 demonstrates that human users can exploit visual patterns that they recognize on the screen. They can define spatial location paths based on layout and DOM structure¹. We argue that in the future machines will be able to perform a likewise step of exploiting visual patterns for learning extraction rules, because the spatial location paths are conceptually simpler than their XPath 1.0 based counterparts.

¹ In this example, the DOM structure in use was restricted to XML node types.

Example 11.3. XPath allows also to navigate and query strongly unstructured presentation-oriented documents.

The PDF document in figure 2.4, can be represented by the simple DOM in Figure 11.2. DOM of PDF documents has the shape of flat trees with all leaf nodes that represent document content items, and a root node that represents the whole document. This structure is built by considering both visualization areas and content items in the PDF stream as described in Section 9.2. The DOM cannot be used to pose query to the document because of its completely flat structure.²

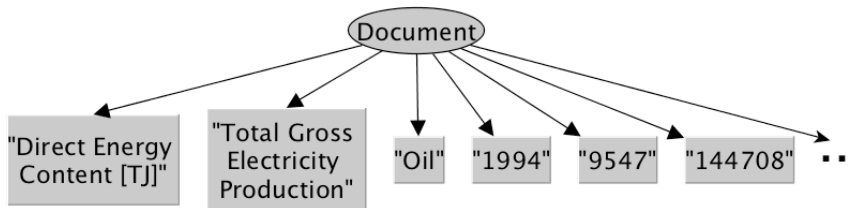


Fig. 11.2. A DOM Portion of the PDF Page in Fig. 2.4

At the contrary, by considering the spatial arrangement of content items in the PDF document suggests to a human reader that s/he is looking at information organized in tabular form. The meaning of each content item in the table comes from the alignment of rows and columns. So the reader interprets strings in the first row and first column as row and column headers and each number in the table body on the base of its headers. XPath, by exploiting spatial information modeled in the SDOM shown in Figure 11.3, allow for spatially selecting nodes of the PDF document.

The following XQuery, based on XPath, allows to acquire table data and store them in structured form. We want to extract row headers, column headers and table values.

```

for $rh in document ("table.pdf")
(3.1) //text [not(W::*)]
return
<table-triples>
{
  for $ch at $j in document ("table.pdf")
(3.2) //text [not(N::*)]
  <row-header>
(3.3) {$rh}

```

² It is noteworthy that a document analysis and understanding process may applied in order to create a more significative DOM. However, it is inherently an inaccurate process and lacks robustness.

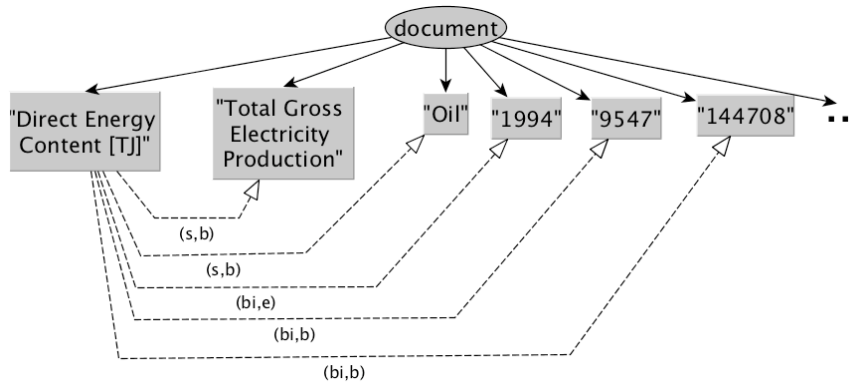


Fig. 11.3. A SDOM Fragment of the PDF Page in Figure. 2.4

```

(3.4)   </row-header>
        <column-header>
        { $ch }
        </column-header>
(3.5)   <value>
        { $rh/E::text [W,$j] }
        </value>
    }
</table-triples>

```

This XQuery allows for acquiring the table in the PDF document in Figure 2.4 as a set of triples of the form `<row-header, column-header, value>`. Firstly spatial location paths 3.1 and 3.2 select row headers (all nodes that have nothing on west) and column headers (all nodes that have nothing on north) respectively. Then, starting from row headers a value related to the j^{th} column header is identified by the spatial location step `$rh/E::* [W,$j]` that navigates step by step nodes on east of row headers by means of position predicates. It is worthwhile noting that spatial arrangement of the PDF document enables to consider the table as a matrix, so the query navigates the table in a very intuitive way.

Example 11.4. In figures 11.4a, 11.4b, and 11.4c friend lists extracted from different social networks pages³ are depicted. Each friend is represented by a photo and its name below the photo. So each friend can be represented by the couple of content items `<photo, name>`. Figures 11.4a and 11.4b also show a simplified DOM where dashed arrows represent some spatial relations expressed in a high level qualitative way. Figures show some interesting aspects

³ <http://www.bebo.com>
<http://my.care2.com>
<http://it.netlog.com>

that are common in HTML documents: (i) Each couple is organized in a single DOM sub tree. (ii) couples in the same DOM are represented by similar subtrees, whereas, in different DOMs couples are represented by subtrees that significantly differ. More in detail, couples `<photo, name>` are represented by: TR and TD tags of an HTML table in Figure 11.4a and DIV tags (suitably nested) in Figure 11.4c UL and LI tags of a HTML list could be also used (Figure 11.4b) as well as more intricate tag structures. (iii) Different subtrees are visualized in the same way. Exploiting only the spatial arrangement, photo and name couples can be captured by the following XQuery based on XPath:

```

for $img in document ("http://www.bebo.com
                      /friendlist.html")
(4.1) //img[ES::*[N,1][self::text]]
      return
        <friend>
          <photo>
(4.2)           {$img}
          </photo>
          <name>
(4.3)           {$img/ES::text[N,1]}
          </name>
        </friend>

```

The spatial location paths 4.1 and 4.3 use the spatial axis ES (Extended at South) that allows for selecting nodes that are at south of a context node and that can be horizontally extended over the south tile of the context node. So *spatial location paths*: (i) 4.1 returns a node set composed of images that have immediately on ES a text (predicate `[self::text]`). In fact, the predicate `[N,1]` selects, among nodes on ES, those that are more close to the image. (ii) 4.3 returns the set of text nodes that are immediately (predicate `[N,1]`) on ES of the image node (the photo). This query can be used for acquiring friend lists from all the web pages shown in Figure 11.4 because it exploits only spatial relation between nodes and is independent from the DOM tree structure.

The Figure 11.5c⁴ shows that: content items that form a single data record (i.e. the couples `<image, description>` in Figure 11.5a) can be split in different sub-trees. In this case each subtree contains the same kind of content items that belong to more data records. Visualization areas, obtained by rendering the document, follow the DOM tree structure that separates images and descriptions in different rows of a table as depicted in Figure 11.5b. Such a situation is a problem for wrapper induction techniques like DEPTA [199] because they recognize data records when they are encoded in the DOM as consecutive similar subtrees. With XPath the previous XQuery can be used, as it is, to obtain the couples `<image, description>`. This example shows

⁴ <http://www.alibris.com/>

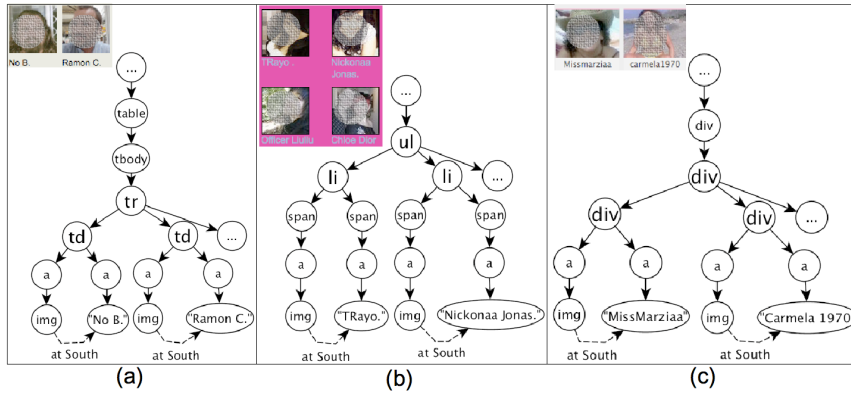


Fig. 11.4. Fragments of Web Pages representing friend lists of social networks (a) Bebo and (b) Care (c) Netlog, with associated DOM structures.

how: (i) information having the same spatial arrangement but different HTML representation can be captured in the same way. (ii) XPath can be exploited in subtasks of wrapper induction techniques. In fact, spatial arrangement of web pages can help in correctly identifying DOM nodes without use of tag nesting.

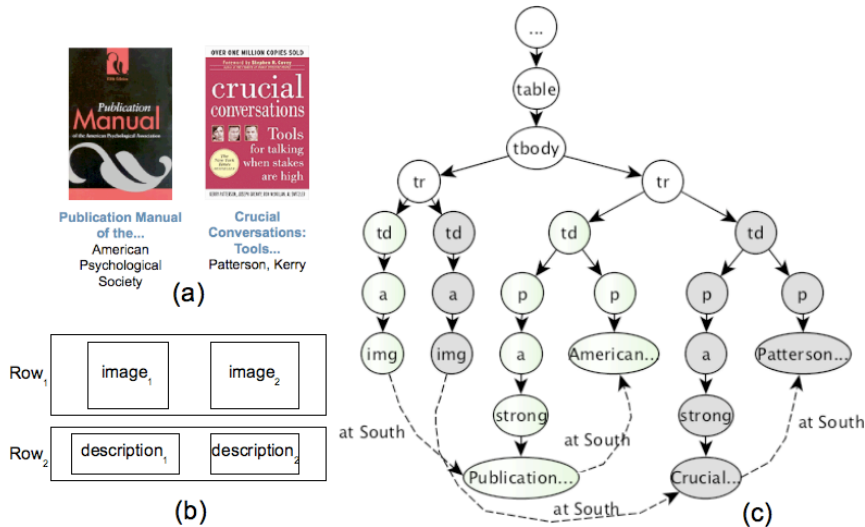


Fig. 11.5. (a) Data items representing books. (b) Visualization areas. (c) DOM tree encoding of data items

11.1.1 Syntax and Semantics

The SXPath language extends the W3C’s XPath 1.0 standard language 2.2 with spatial capabilities. Intuitive navigational features and querying capabilities of XPath 1.0 are central to most XML-related technologies. For this reason XPath 1.0 has attracted great attention in the computer science research community. Even though XPath 2.0 reached recommendation status, it is not well suited for our objectives. In fact, the XPath Core 2.0 query evaluation is PSpace-complete [176, 175, 174, 35] in contrast to the polynomial time complexity of XPath 1.0 [77, 78, 151] (combined complexity). Furthermore, a strong theoretical background on XPath 1.0 is currently available, and specific subsets of the language with attractive properties and essential language features have been characterized [21, 78, 20]. These investigations lay the foundations for understanding the formal properties of SXPath. The SXPath language adopts the path notation of XPath 1.0 augmented by a user-friendly syntax having a natural semantics that enables spatial querying. In particular, SXPath provides: (i) A new set of *spatial axes* that allow for selecting nodes that have a specific spatial relation w.r.t. *context nodes*; (ii) New node set functions, namely *spatial position functions*, that allow for expressing predicates working on positions of nodes in the plane. As already shown in Example 11.2, such extensions are very useful in practice because they enable both end-user to spatial query web documents on the base of what they see on the document, and automatic application to induce a general spatial wrapper, simplifying and upgrading existing approaches (i.e. [96], and [199, 126]). In this section, we first the SXPath spatial capabilities, then we incrementally provide the formal syntax and semantics, and the computational complexity of the language.

Spatial Axes

SXPath allow for navigating the *SDOM* (see Section 9.3). RA relations, stored in the SDOM, represent all qualitative spatial relations between MBRs, but they are too fine grained, verbose and not intuitive for querying. Therefore, for defining SXPath spatial axes we considered the more synthetic and intuitive *Rectangular Cardinal Relation* (RCR) [132] and *Rectangular Connection Calculus* (RCC) [157] models, presented in Section 9.1. Each spatial axes (expressed by a RCR or a topological relation) corresponds to a set of RA relations expressed by the mapping function presented in Definition 9.4, in the Section 9.1.

Like in XPath 1.0, SXPath axes are interpreted binary relations $\chi \subseteq V \times V$ (see Section 2.2.1). Let $self := \{\langle u, u \rangle | u \in V\}$ be the reflexive axis, remaining SXPath axes are partitioned in two sets: Δ_t and Δ_s . The set Δ_t contains *traditional* XPath 1.0 axes (*forward*, e.g. child, descendant, and *reverse*, e.g. parent, ancestor) that allow for navigating along the tree structure. They are encoded in terms of their *primitive* relations (i.e. *firstchild*, *nextsibling* and

their inverses), as shown in Section 2.2.1. The set Δ_s contains the novel (directional and topological) *spatial axes* corresponding to the RCRs and Topological Relations that allow for navigating along the spatial RA relations. In the following we formally define spatial axes in terms of their primitive RA relations stored in the SDOM.

Definition 11.5. *SXPath spatial axes are interpreted binary relations $\chi_s \subseteq V_u \times V_v$ of the following form $\chi_s = \{\{u, w\} \mid u, w \in V_v \wedge mbr(u) \rho mbr(w) \wedge \rho \in \mu(R)\}$. Where R is the RCR or topological relation that names the spatial axis relation and μ is the mapping function.* ■

In order to define the semantics of SXPath we give a trivial generalization of the document total order w.r.t. an axis [184] and define the relative index function.

Definition 11.6. *The document total order w.r.t. the axis χ written as $<_{doc, \chi}$ is: (i) the reverse document order $>_{doc}$ when χ is a traditional reverse axis [184]; (ii) the document total order $<_{doc}$ (Def. 9.9) otherwise.* ■

Definition 11.7. *Let $\Gamma \subseteq V$ be a set of nodes, the index function $idx_\chi(u, \Gamma)$ returns the index of the node u in Γ w.r.t. $<_{doc, \chi}$ (where 1 is the smallest index).* ■

In the following we present the formal syntax of two important fragments of SXPath, namely: *Core SXPath* and *Spatial Wadler Fragment*. The grammar of their unabbreviated version is incrementally provided. Like XPath 1.0, SXPath has also a number of syntactic abbreviations, used in Example 11.2. But they are just syntactic sugar and are not considered in the following.

Core SXPath

We define the fragment of *Core SXPath* as the navigational core of SXPath. It is obtained extending Core XPath [76] (the navigational core of XPath 1.0) by spatial axes introduced in Sec. 11.1.1.

Definition 11.8. *The EBNF grammar of Core SXPath is:*

```

locpath ::= '/' locpath | locpath '/' locpath |
          locpath '|' locpath | locstep
locstep ::= axis '::' t | locstep '[' bexpr ']'
bexpr   ::= bexpr 'and' bexpr | bexpr 'or' bexpr |
          'not(' bexpr ') | locpath
axis     ::= xpathAxis | spatialAxis
xpathAxis ::= 'self' | 'child' | 'parent' | ...
spatialAxis ::= topAxis | dirAxis
topAxis   ::= 'EQ' | 'CD' | 'CR'
dirAxis  ::= 'B' | ... | 'U'

```

where:

- `loxpath` is the start symbol.
- `axis` denotes axis relations that are traditional XPath axis (`xpathAxis`) and atomic directional and topological axes (`spatialAxis`).
- `t` is the node test.

■

Spatial Wadler Fragment

Since Core XPath lacks the ability to exploit spatial position of nodes, we define the Spatial Wadler Fragment (SWF). SWF is the spatial extension of the *Extended Wadler Fragment* (EWF) [77]. It allows positional, logical and arithmetic features.

Definition 11.9. *The syntax of the SWF-Queries is defined by the Core XPath grammar with the following extensions.*

```

expr      ::= locpath | bexpr | nexpr
dirAxis  ::= 'B' | ... | 'U' | disjDirAxis
bexpr    ::= bexpr 'and' bexpr | bexpr 'or' bexpr | 'not(' bexpr ')' |
           nexpr relop nexpr | sexpr relop sexpr | locpath |
           locpath relop sexpr | locpath relop number
nexpr    ::= number | nexpr arithop nexpr |
           'position()' | 'last()' | 'posFromS()' |
           'lastFromS()' | 'posFromN()' | 'lastFromN()' |
           'posFromW()' | 'lastFromW()' | 'posFromE()' |
           'lastFromE()' | 'posSpatialNesting()'
sexpr    ::= string
arithop  ::= '+' | '-' | '*' | 'div' | 'mod'
relop    ::= '=' | '!=' | '<' | '<=' | '>' | '>='

```

where:

- `expr` (instead of `loxpath`) is the start symbol.
- `dirAxis` considers disjunctive directional relation `disjDirAxis`⁵.
- `nexpr` extends traditional XPath numerical expressions with spatial position functions. `number` and `string` denote constant real-valued numbers and strings respectively.

■

We don't give here the syntax of the full XPath language. The reason for this is lack of space. However, by considering [184] extending the syntax is an easy exercise.

⁵ Core XPath does not allow for querying spatial orders, so spatial axes corresponding to disjunction of RCRs do not add expressiveness.

Abbreviated Syntax

As for XPath 1.0, SXPath has an abbreviated syntax. The meaning of abbreviations used in examples is described in Table 11.1.

Full Syntax	Shortcut	Notes
S S:SW S:SE S:SW:SE	<i>ES</i>	Extended at South
W SW:W W:NW SW:W:NW	<i>EW</i>	Extended at West
N NW:N N:NE NW:N:NE	<i>EN</i>	Extended at North
E NE:E E:SE NE:E:SE	<i>EE</i>	Extended at East
[posFromS()=1]	[S,1]	(Similarly for the spatial orders W, N, E, and Containment.)
[posFromS()=2]	[S,2]	
...		
[posFromS()=i]	[S,i]	
[posFromS()=lastFromS()]	[S,last()]	
[posSpatialNesting()=1]	[maximal]	Equivalent to [CD,1]

Table 11.1. Abbreviated syntax

Semantics

In this section we define the semantics of SXPath by adopting the denotational semantics proposed in [185]. Like in XPath 1.0 the main structural feature of SXPath are *expressions*, that return a value from one of the following types: *node set*, *number*, *string*, or *Boolean*. Every expression evaluates relative to a *context*. Context and domain of context are defined in the following as extension of the definition given in [185].

Definition 11.10. *The Context is the following 12-tuple:*

$$\mathbf{c} = \langle n, p_{<_{doc}}, s_{<_{doc}}, p_{\leq_{\uparrow}}, s_{\leq_{\uparrow}}, p_{\leq_{\rightarrow}}, s_{\leq_{\rightarrow}}, p_{\leq_{\downarrow}}, s_{\leq_{\downarrow}}, p_{\leq_{\leftarrow}}, s_{\leq_{\leftarrow}}, p_{\leq_t} \rangle$$

where: (i) n is a context node. (ii) p_{\leq_z} are natural numbers that indicate the context positions w.r.t. orders $\leq_z \in \{<_{doc}, \leq_{\uparrow}, \leq_{\rightarrow}, \leq_{\downarrow}, \leq_{\leftarrow}, \leq_t\}$. (iii) s_{\leq_z} are natural numbers that indicate the context sizes w.r.t. orders in $\{<_{doc}, \leq_{\uparrow}, \leq_{\rightarrow}, \leq_{\downarrow}, \leq_{\leftarrow}\}$. ■

Definition 11.11. *The Domain of Contexts is the following set:*

$$\mathbf{C} = \{ \langle n, p_{<_{doc}}, s_{<_{doc}}, p_{\leq_{\uparrow}}, s_{\leq_{\uparrow}}, p_{\leq_{\rightarrow}}, s_{\leq_{\rightarrow}}, p_{\leq_{\downarrow}}, s_{\leq_{\downarrow}}, p_{\leq_{\leftarrow}}, s_{\leq_{\leftarrow}}, p_{\leq_t} \rangle$$

$| n \in V \wedge 1 \leq p_{<_{doc}} \leq s_{<_{doc}} \leq |V| \wedge 1 \leq p_{\leq_z} \leq s_{\leq_z} \leq |V_v| \wedge 1 \leq p_{\leq_t} \leq |V_v| \}$
where $\leq_z \in \{<_{\uparrow}, \leq_{\rightarrow}, \leq_{\downarrow}, \leq_{\leftarrow}\}$. ■

Definition 11.12. *Let Σ be the labeling alphabet (i.e., “tags”). We define the node test function $T : (\Sigma \cup \{*\} \cup \{text\}) \rightarrow 2^V$ (“node test”) which assigns to each label (XML tag or textual leaf node) the set of nodes labeled with it. Furthermore, $T(*) := V$. ■*

In the following we first define the auxiliary semantic function for location paths then we give the semantics of SXPath.

Definition 11.13. *Location path semantics.* Considering the grammar, Let π, π_1, π_2 be location paths, let $locstep$ be a location step over an axis χ , let $bexpr$ be a boolean expression and let n be a context node, then the semantics function of SXPath location paths

P : LocationPath \rightarrow node \rightarrow nodeset is defined as follows:

$$\begin{aligned}
P[\!/\!\pi] (n) &:= P[\pi](root) \\
P[\pi_1/\pi_2] (n) &:= \{n_2 | n_1 \in P[\pi_1](n) \wedge n_2 \in P[\pi_2](n_1)\} \\
P[\pi_1|\pi_2] (n) &:= P[\pi_1](n) \cup P[\pi_2](n) \\
P[axis :: t] (n) &:= \{n' \mid \llbracket axis \rrbracket (n, n')\} \cap T(t) \\
P[locstep[bexpr]] (n) &:= \{n' \mid \mathbf{W} = P[locstep](n) \wedge n' \in \mathbf{W} \wedge \\
&\quad \varepsilon[bexpr](\mathbf{c}_{n'}) = true \wedge \mathbf{c}_{n'} := \langle w, idx_\chi(n', \mathbf{W}), |\mathbf{W}|, pid_{x_{\leq t}}(n', \mathbf{W}) \\
&\quad plast_{x_{\leq t}}(\mathbf{W}), pid_{x_{< -}}(n', \mathbf{W}), plast_{x_{< -}}(\mathbf{W}), pid_{x_{\leq i}}(n', \mathbf{W}), \\
&\quad plast_{x_{\leq i}}(\mathbf{W}), pid_{x_{< -}}(n', \mathbf{W}), plast_{x_{< -}}(\mathbf{W}), pid_{x_{\leq t}}(n', \mathbf{W}) \rangle\}
\end{aligned}$$

where T is the node test function defined in Def. 11.12, and the semantics of an axis ($\llbracket axis \rrbracket$) is defined as follows:

- $\llbracket spatialAxis \rrbracket := \{(n, n') \mid mbr(n) \rho mbr(n') \wedge \rho = \mu(spatialAxis)\}$ for spatial axes;
- let $\circ, *, +$ denote the concatenation, reflexive and transitive closure of binary relations R_1 and R_2 , respectively, and let $R^{-1} = \{(n', n) \mid R(n, n')\}$ denote the inverse binary relation of a binary relation R . We have:

$$\begin{aligned}
\llbracket self \rrbracket &:= \{(n, n') \mid n = n'\} \\
\llbracket child \rrbracket &:= \{(n, n') \mid n (R_{\downarrow} \circ R_{\Rightarrow *}) n'\} \\
\llbracket parent \rrbracket &:= \llbracket child \rrbracket^{-1} \\
\llbracket descendant \rrbracket &:= \llbracket child \rrbracket^+ \\
\llbracket descendant_or_self \rrbracket &:= \llbracket child \rrbracket^* \\
\llbracket ancestor \rrbracket &:= \llbracket descendant \rrbracket^{-1} \\
\llbracket ancestor_or_self \rrbracket &:= \llbracket descendant_or_self \rrbracket^{-1} \\
\llbracket following_sibling \rrbracket &:= \{(n, n') \mid n R_{\Rightarrow +} n'\} \\
\llbracket preceding_sibling \rrbracket &:= \llbracket following_sibling \rrbracket^{-1} \\
\llbracket following \rrbracket &:= \llbracket ancestor_or_self \rrbracket \circ \llbracket following_sibling \rrbracket \\
&\quad \circ \llbracket descendant_or_self \rrbracket \\
\llbracket preceding \rrbracket &:= \llbracket ancestor_or_self \rrbracket \circ \llbracket preceding_sibling \rrbracket \\
&\quad \circ \llbracket descendant_or_self \rrbracket
\end{aligned}$$

■

Definition 11.14. Given an SXPath expression e and a context \mathbf{c} , the semantics function $\varepsilon : SXPathExpression \rightarrow \mathbf{C} \rightarrow SXPathType$ returns number, string, boolean, or node set values:

$$\begin{array}{ll}
\varepsilon[\pi](\mathbf{c}) := P[\pi](n) & \varepsilon[\text{posSpatialNesting()}](\mathbf{c}) := p_t \\
\varepsilon[\text{position()}](\mathbf{c}) := p_{<doc} & \varepsilon[\text{last()}](\mathbf{c}) := s_{<doc} \\
\varepsilon[\text{posFromN()}](\mathbf{c}) := p_{\leq_1} & \varepsilon[\text{lastFromN()}](\mathbf{c}) := s_{\leq_1} \\
\varepsilon[\text{posFromS()}](\mathbf{c}) := p_{\leq_\uparrow} & \varepsilon[\text{lastFromS()}](\mathbf{c}) := s_{\leq_\uparrow} \\
\varepsilon[\text{posFromW()}](\mathbf{c}) := p_{\leq_\rightarrow} & \varepsilon[\text{lastFromW()}](\mathbf{c}) := s_{\leq_\rightarrow} \\
\varepsilon[\text{posFromE()}](\mathbf{c}) := p_{\leq_\leftarrow} & \varepsilon[\text{lastFromE()}](\mathbf{c}) := s_{\leq_\leftarrow} \\
\varepsilon[\text{Op}(e_1, \dots, e_m)](\mathbf{c}) := F[\text{Op}](\varepsilon[e_1](\mathbf{c}), \dots, \varepsilon[e_m](\mathbf{c}))
\end{array}$$

where:

P is the function in Def. 11.13. (ii) $F[\text{Op}] : O_1 \times \dots \times O_m \rightarrow O$, is defined in Tab. 11.2 (for a complete list of functions we refer to [76, 184]). The meaning of each expression is given by the meaning of its subexpressions. ■

$F[\text{constant number } i : \rightarrow \text{num}]()$	$::= i$
$F[\text{ArithOp: num} \times \text{num} \rightarrow \text{num}](i_1, i_2)$	$::= i_1 \text{ ArithOp } i_2$, where $\text{ArithOp} \in \{+, -, *, /, \%\}$
$F[\text{constant string } s : \rightarrow \text{str}]()$	$::= s$
$F[\text{and: bool} \times \text{bool} \rightarrow \text{bool}](b_1, b_2)$	$::= b_1 \wedge b_2$
$F[\text{or: bool} \times \text{bool} \rightarrow \text{bool}](b_1, b_2)$	$::= b_1 \vee b_2$
$F[\text{not: bool} \rightarrow \text{bool}](b)$	$::= \neg b$
$F[\text{boolean: nset} \rightarrow \text{bool}](I)$	$::= \text{if } I \neq \emptyset \text{ then true else false}$
$F[\text{true():} \rightarrow \text{bool}]()$	$::= \text{true}$
$F[\text{false():} \rightarrow \text{bool}]()$	$::= \text{false}$
$F[\text{RelOp: num} \times \text{num} \rightarrow \text{bool}](i_1, i_2)$	$::= i_1 \text{ RelOp } i_2$, where $\text{RelOp} \in \{=, \neq, \leq, <, \geq, >\}$
$F[\text{RelOp: str} \times \text{str} \rightarrow \text{bool}](s_1, s_2)$	$::= s_1 \text{ RelOp } s_2$
$F[\text{RelOp: nset} \times \text{constant string} \rightarrow \text{bool}](I, s)$	$::= \exists u \in I : \text{strval}(u) \text{ RelOp } s$
$F[\text{RelOp: nset} \times \text{bool} \rightarrow \text{bool}](I, b)$	$::= F[\text{boolean}](I) \text{ RelOp } b$

Table 11.2. Semantics of Functions in SXPath

11.1.2 Complexity Issues

This section summarizes the computational complexity results of the SXPath query evaluation problem. We show that *Core SXPath*, *Spatial Wadler Fragment (SWF)*, and *Full SXPath* allow polynomial time combined complexity query evaluation with increasing degree of the polynomial. In the following theorems we denote by D the XML document, which has size $\Theta(|V|)$, where $|V|$ is the number of nodes of its SDOM representation. It is noteworthy that the SDOM (see Section 9.3) has size $O(|V|^2)$.

Core SXPath Complexity

In [76] it has been shown that Core XPath 1.0 has linear time combined complexity. We extend the Core XPath 1.0 by introducing spatial axes. The following *spatial axis function* returns the set of nodes reached from a set of context nodes along a spatial axis.

Definition 11.15. Let χ_s denote a spatial axis in Δ_s . The spatial axis function, which overloads the relation name χ_s , $\chi_s : 2^V \rightarrow 2^{V_v}$ is defined as $\chi_s(\Gamma) := \{u \mid \exists c \in \Gamma : c\chi_s u\}$, where $\Gamma \subseteq V$ is a set of nodes. Moreover, the inverse spatial axis function $\chi_s^{-1} : 2^V \rightarrow 2^{V_v}$ is defined $\chi_s^{-1}(\Gamma) := \{c \in V \mid \chi_s(\{c\}) \cap \Gamma \neq \emptyset\}$. ■

The *spatial axis function* exploits the following *RA function*:

Definition 11.16. Let ρ denote a RA relation in R_{rec} . The related RA function $f_\rho : V_v \rightarrow 2^{V_v}$ is defined as $f_\rho(u) := \{v \mid u\rho v \wedge u, v \in V_v \wedge \rho \in R_{rec}\}$ ■

The function f_ρ works on the SDOM represented as a nested direct-access data structure. Such data structure for SDOM can be computed in a preprocessing step, which runs in $O(|V_v|^2)$ and requires $O(|V_v|^2)$ more space than a standard DOM. It is noteworthy that f_ρ takes as input a single node $u \in V_v$, accesses in constant time the hash set representing the SDOM, and returns the node set associated to u by the RA relation ρ in constant time. For the function f_ρ the following property holds.

Lemma 11.17. Let ρ be an RA relation. For each pair of nodes $u, v \in V_v$, $u \rho v$ iff $v \rho^{-1} u$.

Now we are ready to define as the *spatial axis function* works:

Let $\Gamma \subseteq V$ be a set of nodes of a SDOM, let $\chi_t \in \Delta_t$ be an axis from the XPath language other than *self*, let $\chi_s \in \Delta_s$ be a spatial axis, let ϱ be a set of RA relations, and let $E(\chi_t)$ denote a regular expression based on the primitives *firstchild* and *nextsibling*, as presented in Section 2.2.1, that define $\chi_t \in \Delta_t$.

The Algorithm 11 computes the set of nodes reached from a set of nodes Γ by means of an axis χ . We distinguish between traditional and spatial axis. So, for traditional axes in (1.1) and (1.2) the algorithm requires time $O(|V|)$ [76]. Whereas, for spatial axis in (1.3) and (1.4), we have to consider the mapping function μ that runs in constant time (see Def. 9.4) and returns a set of atomic/conjunctive RA relations ϱ representing the axis given as input. The function $eval_\varrho$ (1.5)-(1.10), for each input node u and for each RA relation $\rho \in \varrho$ takes the reached nodes using the function f_ρ . The union of the resulting set of nodes (1.9) runs in time $O(|V_v|)$, hence the total time required is $O(|V|^2)$. Now, we are ready to prove the theorem 11.18.

By taking into account previous definition, we can give the following theorem.

Algorithm 11: Axis evaluation

Input: A set of nodes Γ and an axis $\chi \in \Delta$
Output: $\chi(\Gamma)$
Method: $eval_\chi(\Gamma)$

(1.1) **function** $eval_{self}(\Gamma) := \Gamma$.
(1.2) **function** $eval_{\chi_t}(\Gamma) := eval_{E(\chi_t)}(\Gamma)$.
(1.3) **function** $eval_{\chi_s}(\Gamma) := eval_{\{\rho_i | \rho_i \in \mu(\chi_s)\}}(\Gamma)$.
(1.4) **function** $eval_{\chi_s^{-1}}(\Gamma) := eval_{\{\rho_i^{-1} | \rho_i \in \mu(\chi_s)\}}(\Gamma)$.
(1.5) **function** $eval_\varrho(\Gamma)$ **begin**
(1.6) $\Gamma' := \emptyset$;
(1.7) **foreach** $u \in \Gamma \cap u \in V_v$ **do**
(1.8) **foreach** $\rho_i \in \varrho$ **do**
(1.9) $\Gamma' := \Gamma' \cup_{set} f_{\rho_i}(u)$
 od
 od
(1.10) **return** Γ'
end.

Theorem 11.18. Core XPath queries can be evaluated in time $O(|D|^2 * |Q|)$, where $|D|$ is the size of the XML document, and $|Q|$ is the size of the query Q .

Proof. In [77] it has been shown that Core XPath 1.0 has linear time combined complexity by mapping its queries to an algebraic expression. Likewise for Core XPath 1.0 and its extension, every query that falls in the Core XPath can be mapped in time $O(|Q|)$ to an algebraic expression Φ over the set of operations \cap, \cup, \neg, χ (the axis function) and its inverse χ^{-1} and $\frac{V}{root}(\Gamma) := \{x \in V | root \in \Gamma\}$ (that is, returns V if $root \in \Gamma$ and \emptyset otherwise). In the query Q there are at most $O(|Q|)$ of such operations. Each operation in our algebra can be computed in time $O(|V|)$ except for the axis function that is computable in $O(|V_v|^2)$ time bound in the spatial case as shown in Algorithm 11. Hence, the computation has time bound $O(|V|^2 * |Q|)$. ■

Full XPath and SWF Complexity

In this section we first prove the Full XPath query evaluation combined complexity. Then, considering some restrictions, we prove the better complexity bound for the *Spatial Wadler Fragment* (SWF).

In order to obtain a polynomial-time combined complexity bound for Full XPath and then SWF, we use dynamic programming adopting the *Context-Value Table* (CV-Table) principle proposed in [76]. Given an expression e belonging to an input query, the CV-Table of e specifies which value is obtained given a valid context \mathbf{c} : $(\mathbf{c}, \varepsilon \llbracket e \rrbracket (\mathbf{c}))$. The CV-Table of each expression is obtained combining the values of its subexpressions. Moreover, we adopt the

simple idea, first presented in [185], that for evaluating each expression just the necessary information of the context (*relevant context*) can be taken into account. Thus, the CV-Tables can be restricted. The relevant contexts of any expression e associated to a node q of Q can be computed in a preprocessing step as follows: (i) If q is leaf node of the query parse tree, the relevant context depends on e . If e is a constant, then its evaluation does not depend on the input context and thus the relevant context is empty. If the expression is a location path or a positional function, then the relevant context corresponds to the i^{th} value of the context⁶ that defines the semantics of the expression as defined in Def. 11.14. (ii) Otherwise, if e is a location path, then the relevant context of q is the context node. In the other cases, the relevant context of q is given by the union of the relevant context of children parse tree nodes (q_1, \dots, q_k) of q . That is, $RelevantContext(q) := \cup_{i=1}^k RelevantContext(q_i)$.

The context-value table principle avoids exponential time complexity because it guarantees that no evaluation of the same subexpression for the same context is done more than once. So it allows simultaneous evaluation for all possible contexts (node). As in [77] where position and size are computed on demand, we compute all spatial position functions in a loop for all pairs of previous/current nodes. For evaluating SXPath location steps we use the *min context algorithm* presented in [77] with the substantial difference being the computation of location step and spatial position functions. Spatial position functions are effectively used only in predicates of location steps, so the complete context will be needed only for predicates of location steps that use spatial position functions. In the following, for lack of space, we describe only the most costly part of the *location step evaluation algorithm* that computes complete contexts.

The Algorithm 12 corresponds to the semantic definition of location step presented in Def. 11.13. But it considers only the most expensive case that requires the computation of the complete context. Such a case is sufficient for proving the combined complexity of Full SXPath. We need to compute complete context when all expressions $e_1 \dots e_m$, in the input location step, require the evaluation of positions w.r.t. document and spatial orders. As detailed in [77], when expressions do not require evaluation of positions we can pre-compute the CV-Tables because the relevant context is empty or corresponds to a single node. In the instruction (2.2) we obtain the nodes reachable via $\chi :: \tau$. In (2.3)-(2.7) we select nodes that satisfy the predicates $e_1 \dots e_m$. We have to compute the complete context \mathbf{c} corresponding to any pair of previous and current nodes u and w , respectively. The instruction (2.7) builds the context \mathbf{c} that considers the position of the node w in \mathbf{W} w.r.t. the document order and all spatial orders (see definition 11.10). For obtaining spatial ordering we need to apply the *layering* function (instruction (2.6)) to \mathbf{W} (i.e. the set of nodes reached from u). Such a function assigns to each

⁶ Given a context \mathbf{c} then $\mathbf{c}[i]$ represents the i^{th} value of the context (e.g., $\mathbf{c}[3]$ represents the context size w.r.t. the document order).

Algorithm 12: Location step evaluation algorithm

Input: A set of nodes Γ and a location step
 $e = \chi :: \tau[e_1] \dots [e_m]$
Output: $P[[e]](\Gamma)$
Method: $eval(e, \Gamma)$

begin

(2.1) $Res := \emptyset$

(2.2) $W := \chi(\Gamma) \cap T(\tau);$

(2.3) **for each** $u \in \Gamma$ **do**

(2.4) $W' := \{w \mid w \in W \wedge u \chi w\}$

(2.5) **for each** e_i with $1 \leq i \leq m$ (in ascending order) **do**

(2.6) $\mathbf{W} := layering(W')$

(2.7) $W' := \{w \mid w \in \mathbf{W} \wedge \varepsilon[[e_i]](\mathbf{c}_w) = true \wedge$
 $\mathbf{c}_w := \langle w, idx_\chi(w, \mathbf{W}), |\mathbf{W}|, pid_{x_{\leftarrow}}(w, \mathbf{W}),$
 $plast_{\leftarrow}(w, \mathbf{W}), pid_{x_{\rightarrow}}(w, \mathbf{W}), plast_{\rightarrow}(w, \mathbf{W}),$
 $pid_{x_{\downarrow}}(w, \mathbf{W}), plast_{\downarrow}(w, \mathbf{W}), pid_{x_{\uparrow}}(w, \mathbf{W}),$
 $plast_{\uparrow}(w, \mathbf{W}), pid_{x_{\leftarrow}}(w, \mathbf{W}) \rangle\}$

od

(2.8) $Res := Res \cup W'$

od

(2.9) **return** Res

end;

node a layer in order to allow the functions $pid_{x_z}(u, \Gamma)$ and $plast_{x_z}(u, \Gamma)$ (see Def. 9.12) for computing spatial position of w in \mathbf{W} . The layering for each spatial order (directional or topological) can be obtained by applying a topological layering algorithm [55]. For lack of space and because the layering for the other orders can be obtained in similar way, only the layering for the topological order is shown in the following.

Definition 11.19. Let $\Gamma \subseteq V_v$ be a set of SDOM nodes, the topological directed acyclic graph (TDAG) $G_t = (\Gamma, A_t)$ is the graph obtained from the SDOM by considering RA relations that express containment among nodes. So for each pair of nodes n, n' in Γ , an arc is added to A_t iff n' is spatially contained in n . ■

Definition 11.20. Let $\Gamma \subseteq V_v$ be a set of SDOM nodes, and let $G_t = (\Gamma, A_t)$ be the corresponding TDAG, the topological layering $L_t(G_t) = \{l_1, \dots, l_{h_t}\}$ (Def. 9.10) is obtained applying to G_t a topological layering algorithm [55]. ■

Layering algorithm runs in time $O(|\Gamma| + |A_t|)$ by using appropriate data structures with minor modifications to the standard topological sorting algorithm. Topological layers allow for defining a topological ordering among nodes in Γ based on their spatial nesting. So, for example, the first layer represents nodes in Γ that are not contained in other nodes. The second layer

represents nodes that are directly contained in nodes in the first layer at first level of nesting, and so on. The layering for each directional order can be obtained also by using optimized methods that work on a pre-layered version of the SDOM, not explained here for lack of space.

Having obtained the complete context \mathbf{c} , the instruction (2.7) allows for computing the set of nodes W' reached from the current node u and that satisfy the predicates $e_1 \dots e_i$. For the current node w , the value of $\varepsilon[e_i](\mathbf{c}_w)$ is looked up from the table if \mathbf{c}_w exists in the CV-Table of e_i , computed otherwise. The resulting nodeset (instruction (2.9)) is the union of nodes reached from each current node u in Γ and satisfying $e_1 \dots e_m$ predicates.

By taking into account previous definitions and algorithms, we can give the following theorem.

Theorem 11.21. *Full SXPath queries can be evaluated in time $O(|D|^4 * |Q|^2)$ and space $O(|D|^2 * |Q|^2)$, where D is the XML document, and Q is a Full SXPath query.*

Proof. Space complexity. In the preprocessing step we create the SDOM structure, then in order to save the spatial relations $O(|V_v|^2)$ additional space is required w.r.t. the XML document. During the query computation, we know that an input Full SXPath query Q has at most $|Q|$ subexpressions, thus at most $|Q|$ CV-Tables are required. We explicitly set up the CV-table for a subexpression e only if the relevant context of e corresponds to a node (i.e. $\{c[1]\}$), or to the empty set. Hence, the CV-Table has at most $|V|$ rows ($O(|D|)$). Moreover, since Full SXPath and Full XPath 1.0 have the same set of operations and return the same result types, then the most costly operation w.r.t. space size is concatenation on string [77]. We have $O(|D| * |Q|)$ maximal size for one entry value in the CV-Table. Hence we obtain the bound $O(|D|^2 * |Q|^2)$ space combined complexity.

Time complexity. The SDOM computation, which costs $O(|V|^2)$, precedes the query evaluation step. An SXPath expression Q has at most $|Q|$ subexpressions that have to be evaluated. Each subexpression e of the input query Q has to be evaluated for at most $O(|V|^2)$ different contexts that can be computed in a loop over all possible values \mathbf{c} corresponding to pairs previous/current of context-nodes (see algorithm 12). Moreover, it was shown in [77] that time required for computing each XPath operation on any context \mathbf{c} is bounded by $O(|D|^2 * |Q|)$. In our case, we add only spatial relations and the position functions. The former ones, given in input only one context \mathbf{c} can be computed in time $O(|V_v|)$ (In fact, in this case the method $eval_e$ of the algorithm 11, has to run only for a node u (see instruction (1.7)). Given a context \mathbf{c} , a positional function returns the value corresponding to the i^{th} value of the context that defines the semantics of the expression (see Def. 11.14). As shown in Algorithm 12 topological layering is needed (see instruction (2.2) in the Algorithm 12) for computing the complete context, but this operation does not impact the worst case bound. Hence we have the time combined complexity $O(|D|^4 * |Q|^2)$. ■

In order to prove that queries falling in the *Spatial Wadler Fragment* can be evaluated in better time than full SXPath queries (Theorem 11.21), we:

- (i) Consider the syntax defined by the grammar presented in Def. 11.9.
- (ii) Adopt restrictions described in [77] for EWF. Such restrictions imply that functions which select data from the XML document, and boolean expressions that compare node sets are not allowed. Furthermore, boolean expressions of the type `locpath relop sexpr` or `locpath relop number` (see the formal grammar of SWF in Definition 11.9) consider only numbers and strings which size do not depend on the XML document (i.e. are values fixed in the query).
- (iii) Adopt the bottom-up/top-down query evaluation strategy proposed in [77]. Such an evaluation strategy distinguish between *outer* and *inner location paths*, where an inner path appears within a predicate, whereas an outer path does not.

For showing computational complexity of SWF and for lack of space, we describe how the node set resulting for a SWF query can be computed by exploiting the Algorithm 12. In particular, outer location paths are computed by the Algorithm 12 as it is (top-down and forward evaluation). Whereas, inner location paths are bottom-up and backward computed. In this case the evaluation algorithm starts from the last location step in a inner location path, and takes as initial input node set Γ either: all nodes in V (when the considered boolean expression coincides with the inner location path itself), or the set of nodes that satisfy (taking into account above considerations) boolean expressions of the form `locpath relop sexpr` or `locpath relop number` (see the formal grammar of SWF in Definition 11.9). Then evaluation algorithm computes node sets as in the Algorithm 12 where: in instruction (2.2) the axis is χ^{-1} instead of χ (backward evaluation), instruction (2.3) is for each previous $u \in W$, and in instruction (2.4) current node w is in Γ .

Having in mind above discussion we can give the following complexity result.

Theorem 11.22. *SXPath queries that fall in the Spatial Wadler Fragment can be evaluated in time $O(\max(|D|^3 * |Q|, |D|^2 * |Q|^2))$, where D is the XML document, and Q is a SWF query.*

Proof. Time complexity. An SWF query Q has at most $|Q|$ subexpressions that have to be evaluated. Such subexpressions are computed by the Algorithm 12 used as described in the above discussion for enabling both top-down and bottom-up evaluation. Location path evaluation is performed in time $O(|V|^2)$ in instruction (2.2) of the Algorithm 12 (see Algorithm 11). For any expression e , the computation of the result value of e for a single context \mathbf{c} takes at most $O(|Q|)$ time (see restrictions presented above). Furthermore, we have $O(|V|^2)$ contexts (instructions (2.3) and (2.7)) and at most $|Q|$ sub expressions, hence the total time required by each expression e of Q is limited by $O(|V|^2 * |Q|)$. However, let u be the node under evaluation, we have to compute the layering for reached nodes in order to obtain the complete context (instruction (2.6) in the Algorithm 12). The layering method costs $O(|V|^2)$ and has to be computed

for reach node under evaluation. So, we need $O(|V|^3)$ time for computing all contexts (see operations (2.3)-(2.6) in Algorithm 12). Hence, the total computational complexity is bounded by $O(\max(|V|^3 * |Q|, |V|^2 * |Q|^2))$. Since normally, $V \gg Q$ then, the complexity bound follows. ■

Complexity Comparison

The complexity results are summarized in Table 11.1.2. These results are compared with the fragment of XPath 1.0 that they extend.

	XPath 1.0		SXPath	
Space	Core[76]	$O(D * Q)$	Spatial	$O(D ^2 * Q)$
Time		$O(D * Q)$	Core	$O(D ^2 * Q)$
Space	EWf[77]	$O(D * Q ^2)$	SWF	$O(D ^2 * Q ^2)$
Time		$O(D ^2 * Q ^2)$		$O(\max(D ^3 * Q , D ^2 * Q ^2))$
Space	Full[76]	$O(D ^2 * Q ^2)$	Full	$O(D ^2 * Q ^2)$
Time	Xpath 1.0	$O(D ^4 * Q ^2)$	SXPath	$O(D ^4 * Q ^2)$

Table 11.3. Comparison between complexity bound of SXPath and XPath 1.0 for a XML document D and a query Q .

11.1.3 Implementation and Experiments

We have implemented the language in a system that embeds the Mozilla browser and computes the SDOM in real time at each variation of visualization parameters (i.e. screen resolution, browser window size, font type and dimension). This way for each Web page and visualization condition there is a unique corresponding SDOM that enables the user to query the Web page by considering what s/he sees on the screen. For computing the SDOM the system exploits the Mozilla XULRunner⁷ application framework that allows for implementing the function *mbr* (see Def. 9.1) that acquires coordinates of the MBRs assigned by the layout engine to each visualized DOM node. Queries posed by users are parsed and evaluated on the SDOM by the *query evaluator* that returns query answers, and visualizes returned SDOM nodes on the Web page by rectangles with highlighted borders and allows for serializing results as XHTML format.

System Efficiency

For evaluating the SXPath system efficiency we have performed experiments that evidence the practical system behavior for both increasing document and query sizes.

⁷ https://developer.mozilla.org/en/XULRunner.1.9.2.Release_Notes

We have considered: (i) a dummy web page that presents a table with 3 columns and an increasing number of rows; (ii) 3 types of queries, falling in the SWF, based on: (a) traditional axes and position functions, (b) spatial axes (both directional and topological) and spatial position functions; (c) a mix of traditional and spatial axes and position functions. Queries were constructed using the following simple patterns. For query types (a) and (c), the first query was given by: `“/descendant::TD/following-sibling::TD[1]”`. For query type (b), the first query was given by: `“/CD::TD/E::TD[W,1]”`. The $(i+1)$ -th query was constructed by appending the following location paths to the i -th query: (a) `“/following-sibling::TD[1]/preceding-sibling::TD[1]”`; (b) `“/E::TD[W,1]/W::TD[E,1]”`, and (c) `“/following-sibling::TD[1]/W::TD[E,1]”`. For instance, the third query ($i = 2$) for the spatial query type (b) was `“/CD::TD/E::TD[W,1]/E::TD[W,1]/W::TD[E,1]/E::TD[W,1]/W::TD[E,1]”`. All queries aim at extracting the central column of the table in the page. They are based on opposite axes (i.e. “at-east” and “at-west”), so they redundantly jump back and forward within the input documents. Our rationale was that by this way the query processor must handle many different spatial paths in parallel coping thus with an intuitively “difficult” query. We illustrate and compare results for queries falling in SWF.

For evaluating data efficiency of the query evaluator, which is independent from SDOM construction, we have used a fixed SWF query having size $|Q|=167$, (where $|Q|$ is the number of subexpressions in the parse tree of Q). Then, we have computed the needed query time for increasing documents sizes from $|D|=50$ to double the maximum size we found on real-world Web pages, i.e. $|D|=7500$ nodes. Fig. 11.6 depicts obtained curves (in log log scale) indicating polynomial time efficiency w.r.t. document size. In particular, curves slopes are all 1, which indicates linear time behavior. The time needed for constructing the SDOM is depicted as a straight line on the log log scale (see Fig. 11.7). In particular, the line slope is 2 indicating quadratic runtime behavior — as expected. Thus, the runtime requirements for the integrated system with SDOM construction and query evaluation remains polynomial.

For evaluating query efficiency we tested the whole system with fixed document sizes ($|D|=1000$, $|D|=3000$, $|D|=6000$) and increasing query sizes ($i \in [0, 30]$). Fig. 11.8 depicts obtained curves on a log log scale. For this experiment time grows linearly with the query size (curves slopes are all 1). Thus, empirical behavior of the system is better than expected from the SWF worst case upper bound.

Language Usability

In this section we present experiments aimed at assessing the usability of our approach and the enhancements provided by SXPath language over XPath 1.0. For the evaluation we have considered the situation of an expert user aiming at manually developing web wrappers for Deep and Social Web sites. As for this expert we assume that s/he has a good command of XPath. S/he visualizes

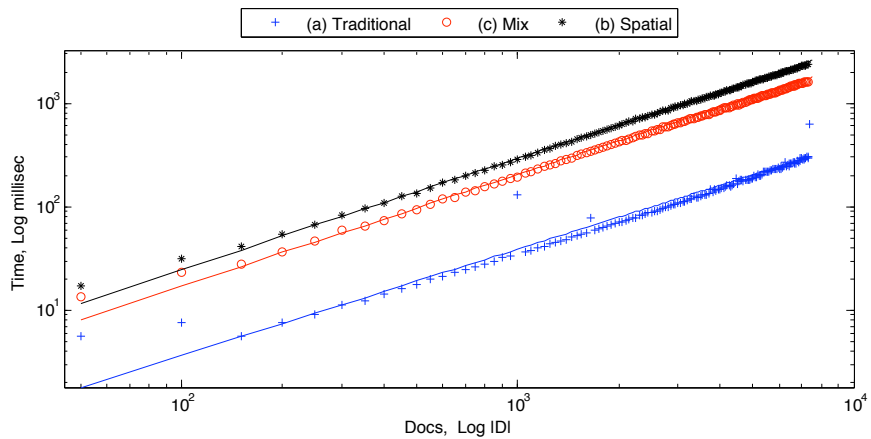


Fig. 11.6. Linear-time Data Efficiency of SXPath Query Evaluation

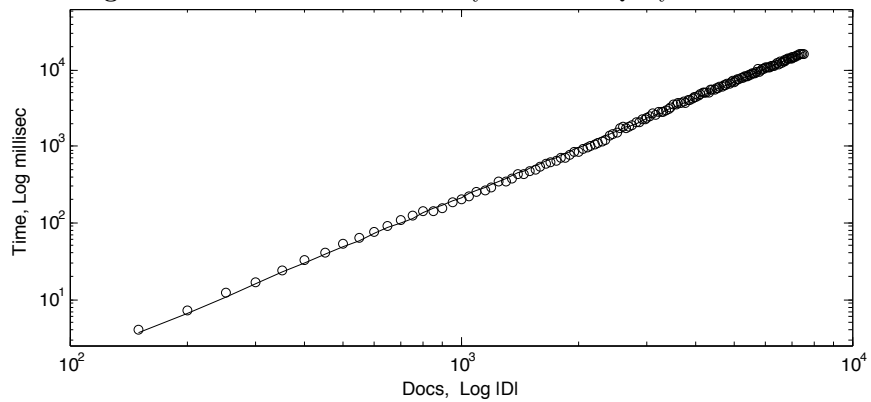


Fig. 11.7. Quadratic-time Complexity of SDOM Construction

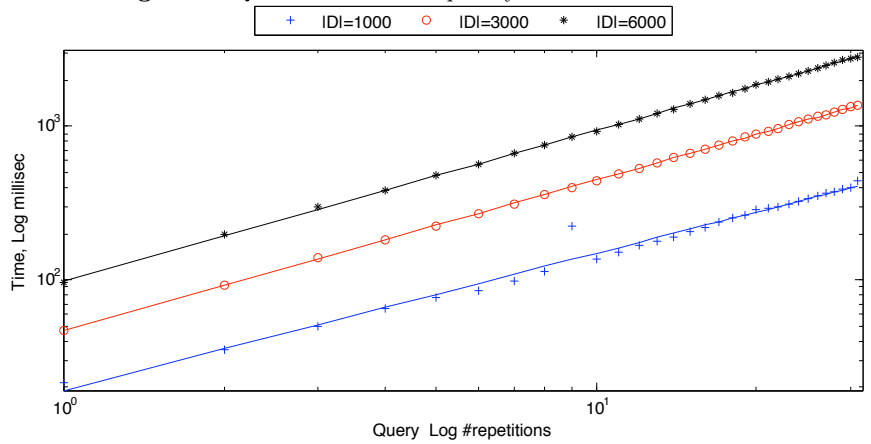


Fig. 11.8. Linear-time Query Efficiency of SXPath Query Evaluation

and explores Web pages by using the XPath system where the embedded Mozilla browser is configured using a typical setting for which Web pages are generally developed (i.e. screen resolution of 1280x800 pixels, browser window size of 1024x768 pixels, standard font type and dimension). We investigate the following questions in our evaluation: (1) How robust is the XPath language w.r.t. Web browser and visualization parameters variations? (2) How “easy” is it for this expert to understand and apply XPath? (3) How suitable is XPath for manual wrapper construction, giving the expert the possibility to look only at the visualized Web page, in comparison to XPath? (4) How suitable is XPath for manual wrapper construction, giving the expert full insight into the SDOM/DOM, when compared against XPath? (5) How transportable is XPath across multiple Web sites in comparison to XPath? Experiments described in the following explore these questions. In the experiments we have used a dataset of 125 pages obtained by collecting 5 pages per site from 25 Deep Web sites already exploited for testing wrapper learning approaches [75, 150, 198] (see Tab. 11.5). Experiments have involved ten users who were students well trained in XPath with no experience in XPath.

Experiment 1. In this experiment we have explored question (1) by evaluating the robustness of XPath w.r.t. Web browser and visualization parameters variations. Firstly, considering the dataset of 25 Web sites listed in Tab. 11.5, we have varied values for screen resolution (i.e. 1280x800, 1024x768, 800x600 pixels), browser window size (i.e. 1024x768, 800x600, 640x480 pixels), font dimension (i.e. small, standard, large, huge), and font type (i.e. times, times new roman, tahoma, arial) and tested if the spatial relations between node change. We have observed that modifications in: (i) Screen resolution and font type do not cause changes in the spatial arrangement of visualized nodes. (ii) Browser window size does not affect 20 Web sites (highlighted by “+” in Tab. 11.5) because they are based on absolute positioning properties, whereas for 5 sites we have changes in the spatial arrangement of visualized nodes. (iii) Font dimension affects the spatial arrangement of visualized nodes. In particular, by using large and huge fonts, visual appearance of Web pages strongly changes w.r.t. standard font dimension. Thus, modifications in screen resolution and font type do not affect query results, whereas changes in browser window size and font dimension could affect the query result. However, this aspect does not impact XPath usability because the XPath system: (i) Embeds the browser and computes the SDOM real time (i.e. at each changing of visualization parameters). So, users can query what they see on the screen at each moment. (ii) XPath queries are stored with visualization parameter settings adopted by the user during the query design process. Thus, when a query is reused on a Web page the embedded browser is set with visualization parameters for which the query has been designed.

Secondly, we have constructed a dataset of 200 Deep Web pages, presenting either data records or tables, randomly selected from the *www.completerlanet.com* web site (the largest portal of Deep Web sites). We have visualized pages by the four most used Web browsers, i.e. Microsoft Internet Explorer,

Mozilla Firefox, Google Chrome, Safari set with default visualization parameters and standard window size (i.e. 1024x768 pixels). We have observed that, even if coordinates returned by different browsers for rectangles bounding visualized nodes were slightly different, the qualitative spatial arrangement of visualized SDOM nodes remained stable. Such a result was expected because rendering rules of most diffused layout engines⁸ have been strongly standardized as it is possible to verify by tests available in [1]. So, XPath queries do not depend on the browser for fixed visualization parameters.

Experiment 2. In this experiment we have investigated question (2) by evaluating the effort needed for learning XPath and the feeling perceived by users in applying the language. We have defined the user task “identify product data records and extract product names and prices” from the web site *www.bol.de*. We have asked users to learn the XPath language and complete the task by writing a sound and complete XPath query looking only at the visualized web page. For learning the language, we have provided users with a short manual explaining spatial axis and position function behavior. We have computed both the number of attempts and the time needed by users for defining the query by using the XPath system. For evaluating the level of “easiness” and “satisfaction” perceived by users in learning and applying the XPath language, at the end of the experiment, we have asked users to answer a questionnaire based on the seven-item Likert scale: very easy/satisfactory (3), easy/satisfactory (2), quite easy/satisfactory (1), neutral (0), quite difficult/unsatisfactory (-1), difficult/unsatisfactory (-2), very difficult/unsatisfactory (-3).

Tab. 11.4 reports results of the experiment: (i) in column 1 the users identifiers; (ii) in column 2 the time needed for learning XPath and manually writing the assigned query; (iii) in columns 3 and 4 answers provided by users to the questions “How easy is the XPath language?” and “What is your level of satisfaction in using the XPath language?” respectively. (iv) in columns 5 and 6 the number of attempts that each user needed for writing spatial location paths for names and prices respectively; (v) in the last two rows the average values and the standard deviations for all observed values.

We have observed that users have required an average number of 4.3 and 3.6 attempts for recognizing The language was assessed as easy to learn and quite satisfactory to use (i.e. average values 2, and 1.2 respectively). Since the system prototype highlights nodes selected by a given XPath query on the screen, users were able to refine the query by visually verifying results for each attempt. For instance, user #9 has written the following spatial location paths `/CD::img/E::text[W,1][N,2]`, and `/CD::img/E::text [N,1][W,2]` for names and prices respectively, by using two attempts for both.

Experiment 3. In this experiment we have explored question (3) by assessing whether spatial information is helpful for a human aiming at manually

⁸ Trident for Microsoft Internet Explorer, Gecko for Firefox, WebKit for Safari and Google Chrome

#user	Time (min)	Easiness/	Satisfaction/	#attempts	
		Difficulty	Unsatisfaction	name	price
1	75	2	0	7	6
2	45	3	2	4	2
3	65	1	1	5	4
4	40	2	1	2	3
5	50	3	2	4	4
6	30	3	3	2	1
7	125	-1	-1	9	8
8	50	2	1	3	4
9	35	3	2	2	2
10	55	2	1	5	2
Average	57	2	1.2	4.3	3.6
σ	26	1.18	1.1	2.2	2

Table 11.4. Evaluation of the Effort Needed for Learn and Apply XPath

writing Web wrappers. We have asked users to perform the same extraction task of Experiment 2 by identifying, for each Web site in the dataset and only by looking at the displayed web pages, the best XPath and XPath queries they could achieve by using at the most 5 attempts.

Table 11.5 reports results: (i) the average number of pairs (product names and prices) correctly extracted (“Cr.”) using XPath and XPath in columns 2 and 5 respectively; (ii) in columns 3 and 6 the corresponding average number of pairs wrongly (Wr.) extracted (false positive – FP/ false negative – FN); (iii) in columns 4, and 7, the average number of attempts performed by users for obtaining the most accurate results (the maximal number of attempts was fixed to 5); (iv) in the last two rows the average recall and precision respectively. Results indicate average precision of 99% and recall of 100% for XPath and average precision of 42% and recall of 99% for pure XPath 1.0. Users were able to define a good XPath query by using 2 attempts on average, whereas all the 5 available attempts were not enough for finding a good pure XPath query for all Web pages in the dataset.

We have investigated the reasons underlying such different levels of effectiveness and have come up with the following observations: (i) XPath queries were constructed by using only spatial constructs that allow for querying visual patterns existing among XML elements that are leaf nodes in the DOM (e.g. `img` and `text` tags) and that help human readers to make sense of document contents; (ii) few errors arise in XPath caused by noise nodes that are visually very similar to the correct ones; (iii) data records in the dataset are constructed by using mainly the tags: `table`, `ul` and `div`. When the tag used in the Web site is `div`, it is very difficult to guess a good pure XPath query;

The experiment clearly shows the advantage of XPath over pure XPath. In fact, only XPath makes spatial layout information explicitly available for querying.

Deep Web Sites	Querying Without DOM/SDOM						Querying With DOM/SDOM							
	XPath			XPath			XPath	Abs. XPath	Rel. XPath	Steps	Att.	Steps	Att.	Steps
	Cr.	Wr.	Att.	Cr.	Wr.	Att.								
amazon.com	58	0/0	2.5	43	32.5/15	5	2.5	5.5	6.5	24	13	10		
bestbuy.com	68	0/2	2.2	70	825/0	5	4.2	4.5	3	12.5	2.5	6		
bigtray.com	125	5/0	2	125	130/0	5	4	4.5	3	10	6	7		
bol.de	60	0/0	2	60	15/0	5	2	4.5	3	16.5	3.5	6.5		
buy.com	100	2/0	3.2	100	30/0	5	4.2	5.5	3	19	4.5	8		
ebay.com	258	0/0	3	258	60/0	5	3	5.5	4.5	21	9	7		
mediaworld.it	125	0/0	2	125	30/0	5	2	5	3.5	21.5	2	5		
shopzilla.co.uk	100	0/0	1.5	100	937/0	5	1.5	4	6.5	23	4	9		
apple.com	50	0/0	1.4	50	0/0	5	1.4	4	4	14	2	4		
venere.com	75	0/0	3.2	75	75/0	5	3.2	4.5	3	9	2.5	4		
powells.com	125	0/0	1.5	125	247.5/0	5	1.5	5	3	11.5	2	4		
barnesandnoble.com	50	0/0	2.3	50	255/0	5	2.3	4.5	3	15.5	2.5	7		
shopping.yahoo.com	75	0/0	1.7	75	187.5/0	5	1.7	4	3	21	3	5		
cooking.com	100	7.5/0	3.2	100	180/0	5	7.2	7.5	8	36	6	9		
cameraworld.com	125	0/0	2	125	10/0	5	2	5.5	3	11.5	2	4.5		
drugstore.com	45	0/4	1.5	41	5/0	5	5.5	8	6.5	16.5	2.5	4.5		
magazineoutlet.com	45	0/0	1	45	125/0	5	1	5	3	21.5	5	9		
dealttime.com	150	0/0	1	150	20/0	5	1	5.5	3	16	3.5	7		
borders.com	125	0/0	1.6	125	40/0	5	1.6	6	3.5	18	4	6		
google.com/products	50	0/0	1.5	50	130/0	5	1.5	5.5	3.5	9.5	2.5	5.5		
nothingbutsoftware.com	80	0.8/0	2.2	80	55/0	5	4.2	6.5	3.5	28	3	8		
abt.com	200	5/0	1.3	200	0/0	5	2.3	6.5	4.5	27	4	6		
cutleryandmore.com	150	1/0	1.5	150	68/0	5	2.5	5	10	36	4	10		
cnet.com	150	0/0	2	130	0/20	5	2	4	4	17.5	5.5	8		
target.com	50	0/0	3	50	2/0	5	3	5.5	3.5	16.5	2.5	5		
Average			2			5	2.7	5.3	4.2	18.9	4	6.6		
Total	2535	27.3/6		2506	3459.5/35									
Recall		100%			99%									
Precision		99%			42%									

Table 11.5. Usability Evaluation of XPath on Deep Web Pages.

Experiment 4. In this experiment we have investigated question (4). We have asked users to perform the same extraction task of Experiment 2 by identifying, for each Web site in the dataset and looking at both visualized Web pages and internal page structures (i.e. DOM and SDOM), sound and complete XPath and XPath queries by using at the most 10 minutes. For writing XPath queries we have provided users by the *DOM Explorer* of the XPath System. DOM explorer visualizes the DOM of a Web page as a tree and allows for selecting a node in the DOM by clicking on an element in the rendered Web page and vice versa. This way users were provided with a visual facility for navigating DOMs of Web pages. This way users were provided with a visual facility for navigating DOMs of Web pages.

Tab. 11.5 gives: (i) in column 8 and 10 the average number of attempts needed by all users for writing sound and complete XPath and SXPath queries respectively. For SXPath queries already sound and complete in Experiment 3 users have not defined new queries, so we have reported in column 8 the value of attempts already observed in column 4. (ii) in column 9, and 11 the average number of location steps in SXPath and pure XPath queries respectively. These numbers has been computed as the average between the number of location steps in location paths that identify product names and prices; (iii) in column 12, and 13 the average number of further attempts and the average number of location steps needed for expressing pure relative XPath queries respectively.

Results (shown in Tab. 11.5) indicate that: (i) For writing queries users needed 4.2 and 2.7 average attempts for pure XPath and SXPath respectively. Since the number of attempts is proportional to needed time, a greater number of attempts indicate that to write queries in pure XPath requires more time in comparison to SXPath. (ii) In defining pure XPath queries all users have shown the tendency to write absolute location paths by deeply analyzing the DOM structure, hence the average number of location steps for pure XPath and SXPath has been 18.9, and 5.3 respectively. Hence, we asked users to write queries by using relative pure XPath location paths by using at most 5 minutes. We observed that further attempts needed by users are proportional to the complexity of the DOM structure.

We have observed that all users have written sound and complete: (i) XPath queries by using more attempts than for SXPath queries; (ii) SXPath queries by adding a pure XPath location path to not sound and complete queries obtained in Experiment 3. XPath Location paths to add have been selected by users looking at the DOM of the pages and choosing the best node to use as data record container. Furthermore, all users have noticed that for some Web sites (highlighted by “ \diamond ” in Tab. 11.5) it has been very difficult to find a sound and complete pure XPath query because of the very intricate DOM structures that make necessary long disjunctions of location paths. Such difficulties are due to the fact that data records in these web sites are contained in discontinuous pieces of the DOM (e.g. www.amazon.com), and that the tag structure representing a given data item can be different from a record to another (e.g. www.ebay.com), even though records have the same spatial arrangement in all selected pages. The experiment shows that, nevertheless both languages have the same effectiveness, writing wrappers for Web pages by pure XPath is more complex than by SXPath. For manually writing pure XPath queries users have needed to deeply analyze the intricate DOM structure for all Web pages, whereas for writing SXPath queries users have mainly looked at the displayed Web pages, this way data records having complex internal structures can be easily and completely described by visually defined manual wrappers. Furthermore, tag structure variety and discontinuities observed in pages DOM are phenomena already known in wrapper induction literature [198, 75]. Such phenomena requires to induce disjunctions, so they

hinder both manual definition of web wrappers and automatic learning of extraction patterns based on XPath.

Experiment 5. In this experiment we have aimed at answering question (5) by evaluating whether SXPath queries are more general and abstract than pure XPath queries given different Web pages and the same extraction task. So, given standard browser settings for all Web sites, we have evaluated whether it is possible to reuse a SXPath query written for a given Web site on other Web sites that present the same information arranged by uniform visual patterns. Firstly, we have considered the subset of Web sites in the dataset that show the same visual pattern for product names. In such Web pages records are vertically listed. Each record is represented by the product image that has at least more than one product attribute, the first attribute from north is the product name. Such Web sites are highlighted by "©" in column 1 of Tab. 11.5. We have observed that the spatial location path `/CD::img[GS|GN::img][GE::*[W,1][N,2][self::text]]/GE::text[W,1][N,1]` is able to identify product names in all these Web sites in a sound and complete way. In contrast, pure XPath location paths for the same subset of sites were all completely different and no reuse of code was possible.

Secondly, we have asked users to write a SXPath query aimed at extracting the list of friends in a social network randomly chosen among those listed in Tab. 11.6. Then we have asked users to try to apply the same query to other remaining social networks in the list. Table 11.6 gives: (i) in column 2 sound and complete queries defined by user #6 without looking at the (S)DOM; (ii) in column 3 the pure XPath location paths that allow for extracting friend names produced by the user looking at the DOM. Even though the internal tag structure of various social networks differ strongly (so different pure XPath queries are needed), all users have been able to use almost the same SXPath query for all social network Web sites. The queries have the following simple basic structure: `/CD::text[.='Amici' or .='Friends']/CR::*[CR,1]/CD::img/GS::*[N,1]`. The SDOM node containing friends list is recognized as the container that either contains the string 'Amici' or 'Friends'. Then friend names are spatially recognized as the first nodes that lie at least in the south tile (spatial axis GS, and spatial predicates [N,1]) of an image. The variety of location paths produced by users, looking at the DOM, and listed in Tab. 11.6 indicates the large heterogeneity of internal tag structures observed for different social networks. This experiment points out that SXPath allows for more general and abstract queries, that are independent from the internal structure of Web pages, in comparison to XPath. Hence, SXPath queries are reusable when information are presented by the same visual pattern.

Experiments provide a strong evidence for believing that humans aiming at manually defining Web wrappers and manipulating Web pages, may benefit from using SXPath navigation instead of pure XPath navigation. Moreover, the transportability of SXPath queries from one Web site to the next simplify manual definition of Web wrappers and can also support wrapper induction from sparsely annotated data, while the lack of such transportability observed

Social Networks	Queries	
	SXPath	Pure XPath
facebook	/CD::text[.="Amici"]/ CR::*[CR,1]/CD::img/GS::*[N,1]	//div[@id='profile_friends_box_inner_content']/div//div/div/div/a
youtube	/CD::text[.="Amici"]/ CR::*[CR,2]/CD::img/GS::*[N,1]	//div[@id='user_friends']//div[1]/div/center/a
netlog	/CD::text[.="Amici"]/ CR::*[CR,2]/CD::img/GS::*[N,1]	//div[@id='nicknameFriends']/div/div/a[2]
care	/CD::text[.="Friends"]/ CR::*[CR,1]/CD::img/GS::*[N,1]	//td[@id='col_right']//table[1]/tbody/tr[1]/td/a[2]
bebo	/CD::text[.="Friends"]/ CR::*[CR,2]/CD::img/GS::*[N,1]	//div[@id='content_Friend']/ul[2]/li/span[2]/a

Table 11.6. Generality of XPath Queries on Social Network Sites

for pure XPath is detrimental for both manual wrapper definition and wrapper induction.

11.2 Spatial Grammars

In this Section we present a query language, named ViQueL, that allows for querying information contained in PODs, on the base of their spatial arrangement. The language is based on context free grammars extended by spatial constructs, named *Spatial Grammars* (SG). A query on a POD is constituted by a set of *spatial production rules* (s-rule) of a SG. Basically s-rules describe how to spatially compose CIs (see Definition 9.6) in order to identify meaningful content structures (e.g. repetitive records, tables, etc.). A SG takes a POD as input, and parses the POD modeled by the 2-D flat representation 9.2. The parsing strategy extends the CYK parsing algorithm (See Section 2.3) so that spatial directions are considered.

In the following, syntax and semantics of the ViQueL language is presented, also by using some running examples. Then, the implemented parsing technique is described, complexity is discussed and experimental results are shown.

11.2.1 Syntax and Semantics

In the following we give by example and intuitive explanation of Spatial Language capabilities and features. Examples describe how users can pose query by considering visual cues that they see on the screen without considering the internal document representation. We will use angular brackets in order to indicate non terminal symbols, whereas uppercase letters for naming terminal symbols. Each terminal symbol of the SG refers to a specific type of CI (see Definition 9.6), whereas each non terminal symbol will specify a Composite CI (see Definition 9.7).

Example 11.23. The Web page shown in Figure 9.6 is a typical answer of Deep Web pages, in which are contained a set of repetitive records. In Figure 11.9 is highlighted one of its records. Dashed rectangles depict CIs and Composite CIs. The spatial arrangement of CIs, and the type of their contents, help human readers to immediately understand the meaning of the record. For identifying and extracting all records in that page, the user could pose the following SG query, expressed in EBNF.

1. $\langle \text{record} \rangle \xrightarrow{W} \text{IMG} \langle \text{description} \rangle$
2. $\langle \text{description} \rangle \xrightarrow{E} \langle \text{textDescription} \rangle \text{TEXT}^+$
3. $\langle \text{textDescription} \rangle \xrightarrow{N} \text{TEXT} \text{TEXT}$

Rule 1 describes the whole record, which is identified by an image and a description on West (W). A description (rule 2) is composed by a textual description and an horizontal sequence (E) of features that are represented by text. A textual description (rule 3) is the name of the product on top of (N) a descriptive text.



Fig. 11.9. Basic and inferred CIs of a data record of the Ebay Web Page (Fig. 9.6).

Example 11.24. Figure 2.6 depicts information in tabular form contained in a PDF document. In the figure each CI is highlighted by using rectangles with solid sides. In the following we show a query that allows for recognizing the table.

1. $\langle \text{table} \rangle \xrightarrow{S} \langle \text{header} \rangle \langle \text{body} \rangle$
2. $\langle \text{header} \rangle \xrightarrow{E} \text{TEXT}^+$
3. $\langle \text{body} \rangle \xrightarrow{S} \langle \text{row} \rangle^+$
4. $\langle \text{row} \rangle \xrightarrow{W} \text{TEXT} \langle \text{values} \rangle$
5. $\langle \text{values} \rangle \xrightarrow{E} \text{PERCENT NUMBER}^+$

The rule 1 recognizes the table as an header on top of a body. Rule 2 identify the header of the table as a sequence of `text`-type CIs. Whereas the body (rule 3) is a vertical sequence of rows. Each row is obtained by composing, along the west direction, a `text`-type CI and numerical values.

Example 11.25. In this example we consider the New York Times Web page shown in Figure 11.10 where rectangles with Solid Sides Highlights CIs that Identify a News. Each Rectangle is annotated by the CI type. Each news is composed by a title in blue text (`bluetext`-type CI), optional names of authors in gray text (`graytext`-type CI), a block of plain text (`text`-type CI), and sometimes an image (`img`-type CI).

Frequently significant information is visualized following specific visual patterns that help to interpret its meaning. A SG query that allows for identifying and extracting most relevant news by means visual patterns is show in the following.

1. $\langle \text{news} \rangle \xrightarrow{N} \langle \text{title} \rangle \langle \text{authors} \rangle? \langle \text{article} \rangle$
2. $\langle \text{title} \rangle \rightarrow \text{BLUETEXT}$
3. $\langle \text{authors} \rangle \rightarrow \text{GRAYTEXT}$
4. $\langle \text{article} \rangle \xrightarrow{W|W:B} \text{TEXT IMG?}$

The rule 1 allows for recognizing a news composed by a vertical sequence of a title, an optional CI (the symbol “?” is used) that represents authors, and the newspaper article. Rule 2 and 3 can be interpreted as `is_a` relations. Rule 4 represent the body of a news that is represented by flat text and optionally by



Fig. 11.10. The New York Times Web page.

an image. The MBR of the image could overlap (W:B) the region of the text. Such a query, written in EBNF, exploits some CI types recognized during the document preprocessing step: **text**-type that represents plain text, **number**-type that represents integer and floating point numbers (**text**-type is more general than **number**-type), **percent**-type that represents percentage and it is a sub type of **number**.

Formally, ViQuEL queries are expressed by spatial production rules (s-rules) of a Spatial Grammar (SG) defined as follows.

Definition 11.26. A spatial grammar is a 5-tuple of the form:

$$SG = (\Sigma, N, S, R_{card}, \Pi)$$

where:

- Σ is a set of terminal symbols that corresponds to CI types and identifies all the CIs the 2-D flat model.
- N is a set of nonterminal symbols that identifies Composite CIs (see Definition 9.7).
- $S \in N$ is the grammar axiom.
- R_{rec} is the set of basic cardinal direction relation introduced in Section 9.1.
- Π is a finite set of spatial production rules (s-rules) of the form $A \xrightarrow{dir} \beta$, where: $A \in N$, $\beta \in (\Sigma \cup N)^+$, and $dir \in (R_{card})^*$.

Σ, N , are disjoint and finite sets . ■

A nonterminal symbol allows the annotation of CCIs. In the current implementation, only s-rules of the form form $A \xrightarrow{dir} \nu\beta$ and $A \rightarrow \nu$, where: $A \in N$, $\nu, \beta \in (\Sigma \cup N)$, and $dir \in (R_{card})^*$ are permitted. It is noteworthy that, without less of generality, rules having as spatial direction a disjunctive RCRs (like

in rule 4 of the example 11.25) are not implemented. They are syntactic sugar, in fact, they correspond to alternative s-rules (e.g. $\langle \text{article} \rangle \xrightarrow{W} \text{TEXT IMG?}$ and $\langle \text{article} \rangle \xrightarrow{W:B} \text{TEXT IMG?}$). Moreover, in the current implementation, in order to compose CCIs, only *spatially continuous* CIs are considered .

Definition 11.27. Let $cci = \langle \Gamma, \alpha \rangle$ be a CCI, where $\alpha = \langle r_{x_\gamma}^-, r_{y_\gamma}^-, r_{x_\gamma}^+, r_{y_\gamma}^+ \rangle$, then the set of CIs contained in Γ are *spatially contiguous* iff α do not overlaps with other CIs not in Γ . ■

A rule of the form $A \rightarrow \nu$, where $\nu \in V$, and $A \in N$ allows for creating a CCI corresponding to a CI identified by the terminal symbol ν . This operation constitutes a generalization of the terminal ν into the nonterminal A , and at the same time a transformation of the CI related to ν into the CCI related to A . Rules of the form $A \xrightarrow{dir} \nu \beta$, where $\nu, \beta \in \Sigma$, and $A \in N$, compose the two contiguous CIs related to the terminal symbols ν and β , along the direction specified by the relation \xrightarrow{dir} , in order to obtain a new CCI A having the structure described in Definition 9.7. S-rules having the form $A \xrightarrow{dir} BC$, where $A, B, C \in N$, compose the sets Γ_B and Γ_C of CIs in the CCIs related to the nonterminals B and C respectively, in order to obtain a new CCI having coordinates computed as specified in Definition 9.7. Similar considerations can be done for rules that combine terminal and nonterminal symbols.

By taking into account the concept of composed content item we can now give the semantic of the ViQueL language in terms of the semantic of a spatial grammar.

Definition 11.28. *Semantic of ViQueL queries.* Let Q be a ViQueL query (i.e. a SG grammar), let σ be a content structure (i.e. a sentence in the language $L(Q)$ of Q), let D be a set of CIs that constitutes the 2-D model of a POD, and let D^* be the "spatial closure" of D (i.e. the set of all possible CCIs that can be obtained by properly composing spatially contiguous CIs in D), the semantic function $P : D \rightarrow D^*$ is $P[Q] := \{\sigma \in D^* \mid \sigma \Rightarrow_Q^* \sigma\}$, where the relation \Rightarrow_Q^* is computed by means of the spatial CYK algorithm. The algorithm works by associating the set Σ of CI types and the set N of CCI identifiers to terminal and nonterminal symbols in Q respectively.

In the following we present the *spatial CYK* (SCYK) algorithm that computes ViQueL queries. SCYK extends the CYK algorithm (see Section 2.3) by: (i) the ability to manipulate two-dimensional CIs and CCIs associated to terminal and nonterminal symbols of the spatial grammar respectively; (ii) some techniques borrowed from Early parsing algorithm. In the following we present the pseudo-code of the SCYK algorithm. In the algorithm the CNF-like normal form is adopted defined as follows (Definition 11.29), because it allows a more intuitive pseudo-code. It is possible and easy to extend the algorithm to parse any type of rules.

Definition 11.29. *A SG is in SG-normal form if all its production rules are either in the form $A \xrightarrow{dir} BC$, or $A \rightarrow \nu$ where A, B and C are non-terminals, while ν is a terminal symbol. Production of type $A \rightarrow \nu$ are called unary spatial production rules.*

The algorithm takes as input a SG Q and a set of CIs D . In instruction 1 the algorithm creates two ordered sets L_x and L_y that contain coordinates r_x^- and r_y^- of all CIs $\in D$ respectively. It is noteworthy that in the worst case sizes $n = |L_x|$ and $m = |L_y|$, can be at most equal to the size of the document $|D|$. But in real cases both $|L_x|$ and $|L_y|$ have a size smaller than $|D|$. In instruction 4 s-rules in Q are acquired in the set RS . Instruction 6 assigns to RS_U all unary s-rules in RS . In instruction 7 non unary s-rules are split in two subsets RS_H and RS_V that contain rules of the type $V \xrightarrow{dir} XY$, where dir is a RCR that expresses basic or multi-tiles relations obtained from the subsets of basic RCR $\{E, SE, NE, W, SW, NW, B\}$ for RS_H and $\{N, NW, NE, S, SE, SW, B\}$ for RS_V (see Section 9.1). Instructions 8 and 9 generate tables T_1 and T_2 respectively. This two tables are handled by using dynamic programming. Indices in the first four positions of T_1 and T_2 , namely i_2, i_1, j_2 , and j_1 identify the CCI having as bottom-left vertex $(L_x[i_2], L_y[j_2])$ and as top-right vertex $(L_x[i_2 + i_1], L_y[j_2 + j_1])$. The last position in table T_1 contains a nonterminal symbol.

The general idea which guides the algorithm is that elements in T_2 represent CCIs by the corresponding coordinates, while elements in T_1 are boolean values that state if a given nonterminal symbol V in the grammar is associated to the corresponding CCI in T_2 (it is noteworthy that a CCI can have different associated nonterminal symbols). Instruction 10 creates a two-dimensional table I where elements $I[i_1, j_1]$ contain a set of couples $\langle i_2, j_2 \rangle$ which indicate that the CCI in $T_2[i_2, i_1, j_2, j_1]$ is not null. Instruction 11 creates the table res that represents the result of the algorithm (i.e. the set of CCIs that satisfies the axiom S in the grammar Q). Instruction 12 initializes tables T_1, T_2 , and I by using the set D of input CIs and the set RS_U of unary s-rules. The initialization procedure works as follows: if an area having as vertices $(L_x[i_2], L_y[j_2])$ and $(L_x[i_2 + i_1], L_y[j_2 + j_1])$ contains only one CI, the couple $\langle i_2, j_2 \rangle$ is added to $I[i_1, j_1]$ and $T_2[i_2, i_1, j_2, j_1]$ is filled by entering coordinates of that CI. Let ν be a CI type (i.e. a terminal symbol), then for each unary rule $V \rightarrow \beta$, where β isa ν (isa is computed by using the taxonomy of CI types), element $T_1[i_2, i_1, j_2, j_1, V]$ is set to *true* (such an operation corresponds to the generation of a CCI for each initial CI). Remaining values in tables T_1, T_2 and I are computed in the main loop (instructions 13-36). CCIs of increasing sizes are discovered by iterating with the two most external cycle by using indexes i_1 (to increase length) and j_1 (to increase height) in instructions 13 and 14 respectively. Two different internal loops are then used in order to find larger CCIs in horizontal and vertical directions respectively, by merging contiguous CCIs. In instruction 15 each iteration uses pairs $\langle i_2, j_2 \rangle$ in the entry $I[i_1, j_3]$ of table I (instruction 16) in order to consider

Algorithm 13: Spatial CYK

Input : A SG Q and a set D of CIs (i.e. the document)
Output: A set of CCIs that satisfy the grammar Q

```

1 <  $L_x, L_y$  > = createOrderedCoordinateSets( $Q$ );
2  $n = |L_x|$ ;
3  $m = |L_y|$ ;
4  $RS = \text{getRuleSet}(Q)$ ;
5  $R = \text{getNonTerminalNumber}(RS)$ ;
6  $RS_U = \text{getRuleSet}(RS)$ ;
7 <  $RS_H, RS_V$  > = splitRuleSetByDirection( $RS - RS_U$ );
8 createTable  $T_1[n, n, m, m, R]$ ;
9 createTable  $T_2[n, n, m, m]$ ;
10 createTable  $I[n, m]$ ;
11 createSet  $res$ ;
12 <  $T_1, T_2, I$  > = initialize( $D, RS_U$ );
13 for  $i_1 \leftarrow 1$  to  $n$  do
14   for  $j_1 \leftarrow 1$  to  $m$  do
15     for  $j_3 \leftarrow 1$  to  $j_1 - 1$  do
16       indexSet =  $I[i_1, j_3]$ ;
17       foreach <  $i_2, j_2$  >  $\in$  indexSet do
18         if  $j_2 + j_1 \leq m$  then
19           for  $i_3 \leftarrow i_1$  downto 1 do
20             verify( $i_2, i_1, j_2, j_1, i_1, j_3, i_2, i_3, j_2 + j_3, j_1 - j_3,$ 
21                $T_1, T_2, RS_V, res$ );
22             end
23           end
24         end
25       end
26     for  $i_3 \leftarrow 1$  to  $j_1 - 1$  do
27       indexSet =  $I[i_3, j_1]$ ;
28       foreach <  $i_2, j_2$  >  $\in$  indexSet do
29         if  $i_2 + i_1 \leq n$  then
30           for  $j_3 \leftarrow j_1$  downto 1 do
31             verify( $i_2, i_1, j_2, j_1, i_3, j_1, i_2 + i_3, i_1 - i_3, j_2, j_3,$ 
32                $T_1, T_2, RS_H, res$ );
33             end
34           end
35         end
36       end
37     end
38   end
39 end
40 return <  $res, T_1$  >;

```

Procedure *verify*($i_2, i_1, j_2, j_1, i_1', j_1', i_2'', i_1'', j_2'', j_1'', T_1, T_2, RS, res$)

```

1  $p_1 = Q[i_2, i_1', j_2, j_1'];$ 
2  $p_2 = Q[i_2'', i_1'', j_2'', j_1''];$ 
3 if  $p_2 \neq null$  then
4   foreach  $V \xrightarrow{dir} XY \in RS_V$  do
5      $c_1 = T_1[i_2, i_1', j_2, j_1', X] \wedge T_1[i_2'', i_1'', j_2'', j_1'', Y] \wedge (p_1 \text{ dir } p_2);$ 
6      $c_2 = T_1[i_2, i_1', j_2, j_1', Y] \wedge T_1[i_2'', i_1'', j_2'', j_1'', X] \wedge (p_2 \text{ dir } p_1);$ 
7     if  $c_1 \vee c_2$  then
8        $T_1[i_2, i_1, j_2, j_1, V] \leftarrow true;$ 
9       update( $T_2[i_2, i_1, j_2, j_1]$ );
10      if  $V$  is axiom then add( $res, T_2[i_2, i_1, j_2, j_1]$ );
11    end
12  end
13  break;
14 end

```

each existing CCI cci_1 having width equals to $|L_x[i_2 + i_1] - L_x[i_2]|$ and height equals to $|L_y[j_2 + j_3] - L_y[j_3]|$. The inner cycle (Instruction 19 and procedure *verify* in instruction 20) takes (if exists) the widest CCI cci_2 contiguous to cci_1 . It is necessary to iterate from i_1 to 1 with index i_3 because we have to consider also inclusion relations (B) between two CCIs. If we don't consider inclusion relation we can take as contiguous CCI cci_2 the one having as corners $(L_x[i_2], L_y[j_2 + j_3])$ and $(L_x[i_2 + i_3], L_y[(j_2 + j_3), (j_1 - j_3)])$. If such CCI exists the algorithm examines each rule $V \xrightarrow{dir} XY \in RS_V$. If values of the two elements in T_1 referring to (cci_1, X) and (cci_2, Y) are *true* and cci_1 is at *dir* to cci_2 then $T_1[i_2, i_1, j_2, j_1, V]$ is set to *true* and value of $T_2[i_2, i_1, j_2, j_1]$ is set to the proper dimension of the CCI. If V is the start symbol in Q , the CCI just found is added to the result set res . Specular operations are done in the second sub-loop in order to evaluate s-rules in RS_H . At the end the algorithm returns the set res containing founded CCIs and the table T_1 that can be used for generating the query (grammar) parse tree by a trace-back procedure.

11.2.2 Complexity Issues and Experiments

The parsing strategy is an ad hoc extension of the CYK parsing algorithm. Despite a considerable expressive power, combined complexity of ViQueL is in P-Time.

Considering that the CYK algorithm has polynomial complexity bound $O(|w|^3 \cdot |G|)$ (See Section 2.3), where $|w|$ is the string size and $|G|$ is number of nonterminals in the grammar, the increased expressiveness of spatial gram-

mars w.r.t. traditional CFG leads only to moderate increase in worst case computational complexity as shown in the following.

Theorem 11.30. *Let D be the SDM of a presentation-oriented document, the evaluation of a ViQueL query Q , performed by Algorithm 13, requires space $O(|D|^4 \cdot |Q|)$ and time $O(|D|^6 \cdot |Q|)$, where $|D|$ and $|Q|$ are the size of the document in terms of CIs and the size of the query in terms of SPRs respectively.*

Proof. Space complexity. Memory usage of the algorithm corresponds to the size of table T_1 . In table T_1 , R represents the number of nonterminals in the query Q , so the space complexity is $O(m^2 \cdot n^2 \cdot R)$. Since the number of nonterminals can be at most $|Q|$, and in the worst case we have that $m = n = |D|$. The space complexity bound $O(|D|^4 \cdot |Q|)$ follows.

Proof. Time complexity. Filling ordered sets L_x and L_y can be done in $O(|D| \cdot \lg(|D|))$. Comparisons in the algorithm can be all done in constant time using appropriate data structures. Remember also that m and n are both bounded by $|D|$. Operations concerning splitting of SPRs in Q are done in linear time, i.e. $O(|Q|)$. Initialization procedure can be performed in $O(|D|^2 \cdot |Q|)$. Main loop is clearly the part of the algorithm that takes more time. Procedure *verify* is manifestly in $O(|Q|)$. The maximum number of couple $\langle i_2, j_2 \rangle$ that can be contained in a element $I[i_1, j_1]$ of table I is $O(m \cdot n)$, in the worst case scenario we have $O(|D|^2)$ because $m = n = |D|$. So time complexity in the worst case is computed as $O(n \cdot m \cdot (m^2 \cdot n \cdot (n \cdot |Q|) + n^2 \cdot m \cdot (m \cdot |Q|))) = O(n^3 \cdot m^3 \cdot |Q|)$. By considering the size of the input document $|D|$ the combined time complexity bound $O(|D|^6 \cdot |Q|)$ follows.

If we don't consider the inclusion spatial relation B (see Section 9.1), then we don't need iterations over n or m before the *verify* procedure. In this case complexity lowers to $O(n^2 \cdot m^2 \cdot (m \cdot |Q| + n \cdot |Q|))$, i.e. $O(|D|^5 \cdot |Q|)$. In the average case the number of couples $\langle i_2, j_2 \rangle$ in $I[i_1, j_1]$ is very low, typically $O(1)$. By these assumptions average case complexity is $O(n^2 \cdot m^2 \cdot |Q|)$, i.e. $O(|D|^4 \cdot |Q|)$, considering inclusion relations and $O(n \cdot m \cdot (m \cdot |Q| + n \cdot |Q|))$, i.e. $O(|D|^3 \cdot |Q|)$, without.

In Figure 11.11 are shown results of experiments carried out for empirically verify Theorem 11.30. We ran experiments on a Windows 7 machine with 2,53 GHz Intel core-duo processor and 4 GB of RAM. We considered the table and the query in Example 11.24 and linearly increased their sizes by adding table rows and dummy rules respectively. Figures 11.11a and 11.11b show required space and time for document sizes that grown from $|D| = 25$ to $|D| = 1020$. Figures 11.11c and 11.11d show required space and time considering query sizes form $|Q| = 10$ to $|Q| = 108$. Curves in the figures refer to normalized values and has been drawn in loglog scale. Experiments show that required space in the average case is linear w.r.t. both the size of Q and D , while required time, in the average case, is linear w.r.t. the size of Q and polynomial (with a degree smaller that in the worst case) w.r.t. the size of D . In Example 2 the system takes about 200 milliseconds for recognizing the table, so the

implemented system results effectively usable in real cases. We have, also, performed usability experiments by asking 10 user to learn the language and apply it for extracting tables and data records from a dataset composed of 5 PDF documents and 5 web pages. Experiments have shown that the language is easy to learn and that can be intuitively applied for real extraction tasks. We don't give further details about usability experiments for lack of space.

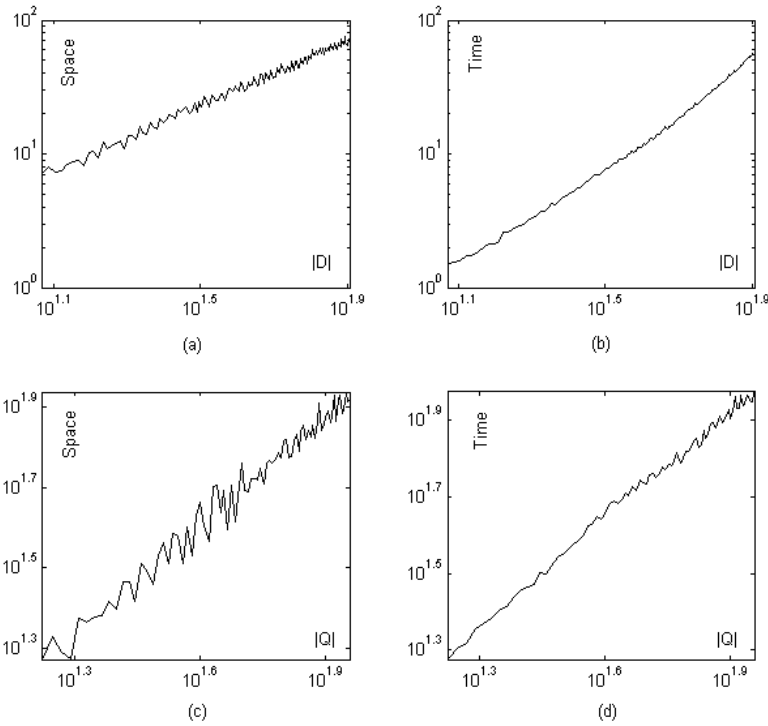


Fig. 11.11. Results of Experiments

11.3 Discussion

In this Section has been presented the SXPath and the ViQueL languages. SXPath extends XPath to include spatial navigation into the query mechanism. ViQueL is founded on spatial grammars that are obtained by combining classical context free grammars and qualitative spatial reasoning constructs. Both languages allow for querying presentation-oriented documents on the base of their visual appearance.

The SXPath language uses navigational primitives based on spatial algebras. Queries are evaluated over the extension of the XML document object model (DOM), i.e. the SDOM. The theoretical complexity of the extended query language has also been evaluated. Furthermore, the theory has been implemented in a SXPath tool. Empirical evaluation has been performed for testing its performances and functionality. Results on representative real web pages have evidenced practical applicability of SXPath. The language can still be handled efficiently, yet it is easier to use and allows for more general queries than pure XPath. The exploitation of spatial relations among data items perceived from the visual rendering allows for shifting parts of the information extraction problem from low level internal tag structures to the more abstract levels of visual patterns. The SXPath query language will be a stepping stone for future work on extracting information from Web pages. Can be argued that SXPath could improve both human and automatic capabilities in querying and extracting information. For instance, the popular LIXTO system [75], which allows users to visually define web wrappers, uses XPath patterns and depends on internal HTML tag structure. It could exploit SXPath in order to leverage visual patterns as desired by authors of [19]. Furthermore, SXPath can be profitably applied in information extraction approaches that make use of pages as rendered by a web browser like the one proposed in [72]. SXPath, in fact, could allow existing wrapper induction and web table extraction approaches (e.g [72] and [198]) to circumvent complex internal representation of web page that hinder extraction rules learning. The new navigational primitives can be used for building of and automated learning of wrappers. In particular, SXPath could be used for: (i) defining novel wrapper induction techniques that learn web wrappers by exploiting the regularity of visual patterns. In particular, SXPath can be seen as a query language for spatial graphs (i.e. the SDOM), so different keygraph searching techniques (see [166] for a survey and [149, 200]) could be applied in order to learn subgraphs that represent data records in a deep web pages. (ii) describing relationships between data entities, such as in this paper, or for co-learning ontology and data extraction, such as discussed in other recent papers [126].

The main feature of ViQueL is that it allows for easily defining visual queries that enable to recognize complex content structures in both HTML and PDF documents. ViQueL queries are computed by the SCYK algorithm, a spatial extension of the well known CYK algorithm. Despite the increased expressiveness of spatial grammars, the complexity of the SCYK algorithm remains in P-Time. Furthermore, experiments prove that the algorithm is efficient and usable in real-life cases with satisfactory results. The proposed approach can be improved by adopting a stochastic extension to SGs in order to better manage ambiguous queries. The ViQueL language can be combined with graph based representation of Web pages (i.e. the SDOM), and with inductive approaches that allow for learning ViQueL wrappers from portions of documents visually selected by users. This way no manual code writing will be required to the users.

Semantic Information Extraction

Nowadays, there is a growing interest in ontologies that are expected to extend current information technologies capabilities. Ontologies enable to directly encode domain knowledge in software applications, so ontology-based systems can exploit the meaning of information for providing advanced and intelligent functionalities, such as information extraction. In this chapter an approach that allows the semantic extraction of information from PODs, named eXtraction ONTOlogies (XONTO), is presented. It enables to use ontologies for: (i) describing in a combined way semantic and syntactic structure of information to extract; (ii) providing the schema in which extracted information can be directly stored during the extraction process; (iii) extracting information from both natural language (flat text) and 2-dimensional spatial arrangement, such as tables, infoboxes, etc.

The ontology-based system XONTO is founded on the idea of *self-describing ontologies*(SDOs), which bring together : the expressive power of ontologies, the spatial grammar with attribute formalisms and the 2-dimensional document model. A SDO is an ontology in which objects and classes can be equipped by a set of rules, named *descriptors*, Descriptor are *object-oriented* spatial grammar rules that *describe* how automatically recognize and extract ontology objects contained in PODs, also when information is arranged in tabular form. Thus a SDO, expresses the semantic of the information to extract and the rules that, in turn, populate itself. In order to allow for extracting information having valuable spatial arrangement, such as tabular form, the SDO takes as input PODs represented by the 2-D document model (see Section 9.2).

In order to yield the actual semantic of a SDO, i.e. recognize and extracting information, two approaches are used: (i) The first approach, implemented in XONTO-L system, exploits the logic representation of ontologies and encodes the spatial grammar into logic rules, and then infer new objects by means the reasoning process. It has been published in [137, 139, 143]. (ii) The second approach, implemented in XONTO-G system, extends existing efficient pars-

ing algorithms for CFGs, in order to parse by means descriptors PODs taken as input. It has been published in [138].

The remainder of this chapter is organized as follows. Section 12.1 describes the SDO approach and shows a running example. The XONTO-L and the XONTO-G system are described in the Section 12.2 and 12.3 respectively.

12.1 Self-Describing Ontologies – SDO

The XONTO system implements the *self-describing ontology (SDO)* approach. In the following the formal structure and the syntax of *SDOs* is presented. To facilitate *SDO* approach understanding a running example regarding the extraction and storing of information on weather, coming from the page depicted in Figure 10.1, is described. Weather information are collected and used by different scientific institutions to study forecast models and weather changes at global level.

Definition 12.1. *A SDO is a triple:*

$$SDO = \langle \mathcal{O}_z, \Delta, \delta \rangle$$

where:

- $\mathcal{O}_z = \langle D, A, C, R, I, \preceq, \sigma, \varphi, \iota \rangle$ is an ontology (see Section 2.4.1);
- Δ is a the set of descriptors and it is expressed by the tuple $\langle SG, Attr, Func, Pred \rangle$ that represents an attribute grammar (see Section 2.3) in which $SG = \langle \Sigma, N, S, R_{card}, \Pi \rangle$ is its underlying Spatial Grammar (SG). A SG is defined in Definition 11.26.
- $\delta : I_c \cup C \rightarrow \Delta$, where $I_c \in I$ are class-instances, is the descriptor function that associates an object i in I_c or a class c in C with its descriptor. In particular, $\delta(i)$ is called object descriptor while $\delta(c)$ is called class descriptor.

■

The ontology \mathcal{O}_z is obtained extending Datalog by object-oriented constructs, such as, *class*, *object* and *inheritance*, see Section 2.4.1.

The peculiarity of the SDO paradigm is that elements \mathcal{O}_z and Δ are strictly coupled. In fact, in *SDO* classical attribute grammars has been extended by means of object-oriented features and 2-dimensional capabilities, as follows.

- Sets N and $Attr$ in Δ coincide with the set of classes $(C \cup I)$ and of of attributes (A) in \mathcal{O}_z respectively. That is, *nonterminal symbols* correspond to *oid* and *class names* defined in \mathcal{O}_z .

- Let $a \in Attr(Y)$ be an attribute of a nonterminal symbol $Y \in N$, then t is the type of a if and only if such type is specified in the schema of the class Y , that is $\langle a, t \rangle \in \sigma(Y)$. That is, when a nonterminal symbol correspond to a class name in \mathcal{O}_z its *attributes* are constituted by the attributes of the corresponding class defined in \mathcal{O}_z . When a nonterminal symbol correspond to an *oid* no attributes are defined.
- Only *synthesized attributes* (see Section 2.3) are used in *Attr*, in fact attribute values of a nonterminal in the head of a descriptor are derived only considering values of nonterminal in the body.
- *Funcions* have the form of an instruction in imperative programming. Each function assign a value (i.e. *oid* or data-type value) to a variable that, in the left-side, represents a synthesized attribute. Set *Func* contains *arithmetic expressions, string expressions, list expressions*.
- *Predicates* are constraints, on the attributes values of nonterminal symbols contained in the right-side, expressed by means of *queries* on \mathcal{O}_z . Set *Pred* contains: (i) *comparison predicates* on integers, strings or lists; (ii) *decision queries* on \mathcal{O}_z^+ that extend \mathcal{O}_z by new instances (generated during the extraction process); (iii) a default predicate (associated to each production) checking whether computed attribute values are consistent with respect to \mathcal{O}_z^+ .
- Production rules are expressed in EBNF notation.

SDO approach is obtained by extending the DLP+ ontology representation language (see Section 2.4.1) by means of *object* and *class descriptors*. DLP+ enables to represent an ontology, descriptors allow to recognize ontology objects within PODs, extract and directly store them in the ontology itself. Descriptors exploit some advanced constructs that enable to deal with 2-dimensional information arrangement (i.e. tables). The XONTO system beside the complete and expressive ontology representation language DLP+, uses also powerful reasoning capabilities over represented knowledge provided by the DLV+ system [109, 158]. A *SDO* allows to represent extensional and intensional aspects of domain knowledge by means of ontology *schemas* and their *instances*.

12.1.1 Logic 2-D Representation

In order to allow the process that yields the actual semantic information extraction provided by the XONTO system, the 2-D model of PODs, which also supply the full-text processing results (i.e. MBRs, strings, tokens, NL properties, presentation features etc. – see Section 9.2), is directly stored as a part of the *SDO* instances itself. In the following default *SDO* schemas used to internally represent objects directly obtained from the input document and some examples of instances are shown.

MBRs and 2-dimensional strings, obtained as output of the pre-processing of PODs documents described in Section 9.2, are stored in the *SDO* by means of classes `mbr`, and `maxString`.

```

class mbr(contains:maxString).
    @2:mbr(@2ms).
    @4:mbr(@4ms).
    ...
class maxString(str:string).
    @2ms:maxString(str:"South at 8.0 mph").
    @4ms:maxString(str:"$ Tornado Warning").
    ...

```

Class `mbr` gathers objects representing not overlapped MBRs obtained during pre-processing. The attribute `contains:maxString` of the class `mbr` allows to express which is the string contained in a given MBR. Class `maxString` collects strings contained in MBRs.

```

class element()
class token(txt:string) isa {element}.
    @68torW:token(txt:"Tornado").
    ...
class image(uri:string) isa {element}.
    @78imTorW:image(uri:"C:\figures\tornadoWarning").
    ...

```

Class `token` collects objects representing: (i) sub-strings recognized during the pre-processing by matching, on strings, regular expressions defined in the SDO; (ii) tokens obtained by applying the tokenizer of a NLP tool on strings. Class `image` collects images contained in the input document, the attribute `uri:string` refers the physical position where the image has been stored during the pre-processing.

The semantic IE approach implemented in the XONTO system exploits general linguistic knowledge coming from already existing named entity thesaurus and semantic networks (e.g. WordNet [63]), or created using a GUI (e.g. regular expressions inherent a given knowledge domain can be defined and classified by hand). Such a linguistic knowledge is represented and stored in special areas of a SDO, named dictionaries, having as root the class `dictionaryPattern`. Just a small taxonomy of a dictionary that allows to recognize meteorological information is sketched in the following.

```

class dictionaryPattern(regex:string,mode:integer,delimiters:integer).
class dNumber() isa {dictionaryPattern}.
class dFloat() isa {dNumber}.
    float_001:dFloat("[+]?\\d+(?:\\.\\d+)?",2,1).
    ...
class unitOfMeasure(symbol:string) isa {dictionaryPattern}.
class temperatureUnit() isa {unitOfMeasure}.
    fah_001:temperatureUnit("Fahrenheit"| "F").
    ...
class velocityUnit() isa {unitOfMeasure}.
    mph_001:velocityUnit("miles per hour|mph",2,1,"mph").
    ...

```

```

class fractionUnit () isa {unitOfMeasure}
    percent_001: fractionUnit("%|perc(?:ent)?",2,1).
    ...
class dCardinalPoint() isa {dictionaryPattern}.
    n_001:dCardinalPoint("north|n",2,1,"N").
    ...
class dMeteorologicalWord() isa {dictionaryPattern}.
    cloudy_001:dMeteorologicalWord("cloudy",2,1).
    clear_001:dMeteorologicalWord("clear",2,1).
    cloud_001:dMeteorologicalWord("cloud",2,1).
    ...
class dWeatherWarning() isa {dictionaryPattern}.
    torn_001:dWeatherWarning("tornado warning",2,1).
    ...
class dCity() isa {dictionaryPattern}.
    chicago_001:dCity("chicago",2,1).
    ...

```

In the class `dictionaryPattern` attributes `regex:string`, `mode:integer` and `delimiters:integer` respectively represent regular expression used to identify the dictionary entry, matching mode (default value 2 is the case insensitive mode) and the flag delimiters (default value 1 means that standard token-delimiters are used to match a regular expression on document strings).

In the XONTO internal document representation the relation `defBy` associates, as shown in the following, regular expression patterns that are instances of the class `dictionaryPattern` to tokens recognized by matching related regular expression.

```

relation defBy(token:token, regex:dictionaryPattern).

```

Linguistic and presentation features of text are stored in the following *SDO* relations which tuples are created at pre-processing time by exploiting a NLP tool.

```

relation hasNLProperties(tok:token, lemma:string, pt:pos_tag).
    hasNLProperties(tok:@68torW, lemma:"tornado", pt:NN).
    ...
relation hasStyle(tok:token, style:charStyle).
relation hasLink(e:element, uri:string).

```

For each token obtained from the pre-processing: (i) Relation `hasNLProperties` stores its lemma and POS-Tag computed by analyzing 2-dimensional strings by means of an existing NLP tool [34]. The hierarchy `pos_tag` allows, for example, to describe verb tense and plurals associated to tokens. The hierarchy reproduces the standard Penn Treebank tag set. (ii) Relation `hasStyle` stores presentation features by using an internal representation of styles which class `charStyle` is the root. (iii) Relation `hasLink` stores the URI when the image or the token has a link in the document. In the following `pos_tag` and `charStyle` hierarchies are shown.

```

class pos_tag().
  class noun(num:string) isa {pos_tag}.
    NN: posTag (num:"singular").
    NNS: posTag(num:"plural").
  class verb(tense:string) isa {pos_tag}.
    VB: verb(tense:"base form").
    VBD: verb(tense:"past tense").
  ...

class charStyle().
  class fontType(name:string) isa {charStyle}.
    arial:fontType(name:"Arial").
    tnr:fontType(name:"Times New Roman").
  ...
  class fontStyle(name:string) isa {charStyle}.
    bold:fontStyle (name:"Bold").
    italic:fontStyle (name:"Italic").
  ...
  class fontSize(ptValue:float) isa {charStyle}.
    fs12:fontSize (ptValue:12).
    fs14:fontSize (ptValue:14).
  ...
  class colour(hexValue:string) isa {charStyle}.
    red:charColour(hexValue:"FF0000").
    blue:charColour(hexValue:"0000FF").
  ...

```

Furthermore, to exploit coordinates during the reasoning process, the XONTO system stores positions (spatial coordinates) of objects extracted at pre-processing time by the following set of predefined relations.

```

relation mbr_coord_2d(b:mbr, x1:integer, y1:integer, x2:integer,
  y2:integer).
  mbr_coord_2d(@04,1,32,4,33)
  ...
relation token_coord_1d(o:token, ms:maxString, char1:integer,
  char2:integer).
  token_coord_1d(@68torW,@4ms,2,9)
  ...
relation image_coord_1d(o:image, ms:maxString, char1:integer,
  char2:integer).
  image_coord_1d(@78imTorW,@4ms,0,1)
  ...

```

MBRs hold as described in Section 9.2 hold 2-dimensional coordinates. Whereas, tokens and images are strings stored in a `maxString`, their positions are defined by the positions of the initial and final character in the `maxString`.

12.1.2 Object and Class Descriptors

The peculiarity of *SDO* is the ability to express grammatical rules describing patterns of ontology objects and classes that allow for automatically recognizing ontology objects in SDOs. Roughly speaking, descriptors are rules that *describe* how information can be extracted from PODs, and stored as ontology objects. The main important feature of descriptors is that they can exploit each other in describing concepts, so each ontology object can be described by the composition of other objects. This mechanism reproduce natural language behavior in which terms having a given meaning (semantics) are combined, by means of the grammatical rules of a given language, for expressing more complex concepts.

Descriptors are production rules where the right-side (*descriptor-body*) constitutes an (*extraction*) *pattern* which recognition in a document means that the object in the left-side (*descriptor-head*) either: exists in the document (*object descriptor*) or can be extracted and stored as a class instance (*class descriptor*).

As defined in Definition 12.1, descriptors represent attribute grammars extended by means of 2-dimensional composition capabilities (because they exploit Spatial Grammars capabilities, Definition 11.26, and furthermore short-cut constructs are provided) that enable to recognize and extract information having also tabular form.

From a syntactical point of view, a descriptor $d \in \Delta$ has the following form: $\mathbf{h} \xrightarrow{dir} \mathbf{b}$, where *dir* represents a direction as described in SG definition (Definition 2.3), or a special construct \square that allow for expressing tabular arrangement of objects, or a topological containment relation \mathcal{CD} . In particular, using \square , the object in the head constitutes spatially a grid where each row (or column) has the structure represented by the body \mathbf{b} . Whereas, $\mathbf{h} \xrightarrow{\mathcal{CD}} \mathbf{b1} \ \mathbf{b2}$ expresses that the object represented by the nonterminal $\mathbf{b2}$ is spatially contained in the object represented by the nonterminal $\mathbf{b1}$.

In the following are presented a subset of descriptors needed for semantically extracting the table contained in the document shown in Figure 10.1. The object **sterling** is recognized in an input document when a specific token defined by the regular expression **sterling_001** is identified. The corresponding descriptor follows:

$$\langle \mathbf{sterling} \rangle \rightarrow \langle \mathbf{T:token()}, \mathbf{defBy(T,sterling_001)} \rangle. \quad (1)$$

Object descriptors like (1) allow for recognizing the already declared instance of the class **city**. When the dictionary instance **sterling_001** is matched in the document the related object is recognized in the same position. It is noteworthy that descriptors having *oid* in the head (object descriptors) act just as recognition rules (object attributes are already set). Whereas, descriptors having a class name in the head can either create new objects by setting attribute values on the base of document contents, or recognize objects already declared for the class.

The following descriptor allows to extract instances of the class `temperature` described as a sequence of number and unit of measure of temperature.

```
class temperature(value:float,unit:string).
  <temperature(V,U)> -> (2)
    <T:token(S),defBy(T,X),X:dNumber()>{V=#str2float(S);}
    <T:token(),defBy(T,X),X:temperatureUnit(symbol:S)>{U:=S;}.
```

The following class descriptor allows to recognize a percentage by using the dictionaryPattern `percent_001`.

```
class percentage(value:float).
  <percentage(value:V)> -> (3)
    <T:token(S),defBy(T,X),X:dNumber()>{V=#str2float(S);}
    <T:token(),defBy(T,percent_001)>.
```

Weather descriptions are recognized by using the following class descriptors that exploit NLP capabilities.

```
class weatherTerm(term:string)
  <weatherTerm(WT)> -> (4)
    <T:token(S),hasNLProperties(tok:T,p:"JJ")>{WT:=S;}?
    <T:token(S),hasNLProperties(tok:T,lemma:L),
      dMeteorologicalWord(regex:L)> {WT:=WT+S;}.
```

```
class weatherDescription(descr:string).
  CD
  <weatherDescription(D)> --> (5)
    <MS:maxString(S)>{D:=S;} <T:weatherTerm()>.
```

The descriptor `weatherTerm` is a general linguistic pattern which means that a “weather term” is a string composed by an optional adjective followed by a meteorological word (e.g. “Mostly Cloudy”, “Clear”). The descriptor `weatherDescription` represents the whole string of a MBR which *contains* weather terms. As mentioned above, this kind of production rules ($\overset{CD}{\rightarrow}$) check spatial containment between the first and the second object. On flat text, such productions, act essentially as a substring operator.

To recognize the table and store it in structured form the following descriptors are needed. The first descriptor describes how is composed a record, the second defines a table as a sequence of such records.

```
class weatherRecord(wCity:city,wWarns:warnings,wTemp:temperature,
  wHumid:percentage,wPress:pressure,wDescr:weatherDescription,
  wWind:wind).
  EE|ES
  <weatherRecord(C,Wa,T,H,P,D,Wi)> ----> (6)
    <X:city()>{C:=X;} <X:warnings()>{Wa:=X;}
    <X:temperature()>{T:=X;} <X:percentage()>{H:=X;}
    <X:pressure()>{P:=X;} <X:weatherDescription()>{D:=X;}
    <X:wind()>{Wi:=X;}.
```

```

class weatherTable(records:[weatherRecord]).
    [EE|ES]
    <weatherTable(R)>----->
    <X:weatherRecord()>{R:=@addLast(X);}+.

```

(7)

A `weatherRecord` is the sequence of objects described in the body. The descriptor is obtained by using the symbol $\xrightarrow{EE|ES}$ which means that 2-dimensional coordinates of objects must be considered in constructing a record. When is 1-dimensional (i.e. it is recognized as a substring), the MBR that contains it, is automatically considered. Furthermore a record is recognized either as horizontal or vertical objects sequence.

The descriptor `weatherTable` describe a table as a sequence of one or more records in either horizontal or vertical direction. The construct $\xrightarrow{[EE|ES]}$, that is the combination of the symbol `[]` and the directions `EE|ES`, assures that the table is obtained by concatenating (either horizontally or vertically) records that have the same internal structure (i.e. records composed of objects of the same types in the same order). So such a kind of descriptors allows to extract a table and its transposed analogous. Other constructs can be used to enable the recognition of tables, also when they contains unknown objects and/or null values.

It is worthwhile noting that by using only 1-dimensional coordinates the system allows to recognize and extract objects contained in flat text by exploiting also NLP capabilities. While by adopting 2-dimensional coordinates the system allows to recognize tabular presentation of information. Thus, by combining 1 and 2-dimensional capabilities complex information arrangement can be managed.

12.2 XONTO-L: Logic-Based System for Extracting Objects

The logic-based approach of XONTO (XONTO-L) implements a compiling and reasoning strategy that exploits the logical 2-D representation of PODs and the SDO opportunely rewritten in terms of logical programs (see Section 2.4) in order to yield the actual semantic and extracting information from PODs. In the following the prototypical implementation of the XONTO-L system is described. The prototype XONTO-L is built on top of the DLV+ (see Section 2.4.1) system that allows to represent *SDOs* and to yields the actual information extraction from PODs by executing logic programs that implement descriptors. As shown in Figure 12.1 the system prototype, that implements the *SDO* approach, is constituted by the following modules: *Document Analyzer*, *Descriptors Compiler*, *Rules Selector* and *Reasoning Engine*.

In the XONTO-L system the *SDO* taked as input is compiled by the *Descriptors Compiler* in DLP+ object-oriented logic rules. An IE process is

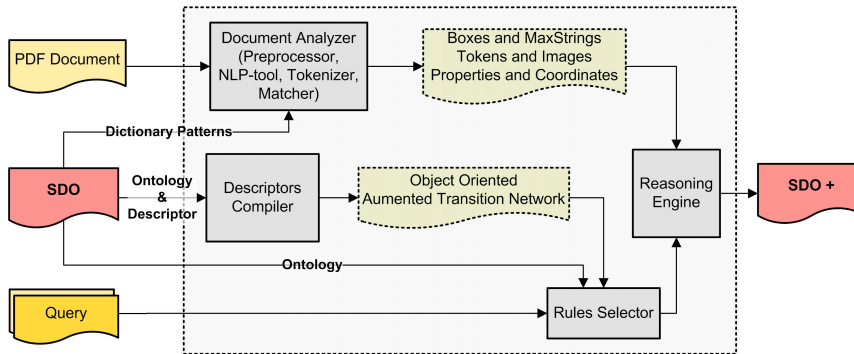


Fig. 12.1. Architecture of the XONTO System

composed by the following steps: (i) PODs pre-processing. In this step the XONTO internal document representation of PODs shown in Section 12.1.1 is obtained by the *Document Analyzer*. (ii) Query analysis and rules selection. In this step the user ask for the extraction of one or more concepts from PODs. The input query, the ontology and the logic program encoding descriptors are analyzed be the *Rules Selector* that provides the minimal. (iii) Reasoning. In this step rules are applied on the input documents. Reasoning result is a *SDO+* that is the original *SDO* augmented by concept instances extracted from the input document. It is worthwhile noting that a *SDO* can be used for many extraction processes where are used different documents and/or queries. In the following the behavior of each system module is described in detail.

Document Analyzer

The *Document Analyzer* takes in input a POD (e.g. a PDF document) and dictionary patterns contained in the *SDO*. The POD is analyzed in order to construct the 2-D Representation, described in Section 12.1.1. In particular, the document analysis consists in: (i) the identification of CIs (ii) the matching of dictionary pattern on obtained strings; (iii) the extraction of presentation features directly acquired by using HTML node attributes or PDF metadata; (iv) the tokenization of available strings in CIs and the acquisition, for each token, of linguistic features (i.e. POS-Tag and lemma) by means of a NLP tool.

Descriptor Compiler

The *Descriptor Compiler* takes in input a *SDO* and rewrites all the descriptors in term of DLP+ object-oriented logic rules. This section describes foundations of the adopted compiling process.

To understand Descriptor Compiler behavior the first aspect to take into account is that the set of descriptors of a *SDO* constitutes an attribute CFG

extended by 2-dimensional capabilities. Since the backbone of descriptors is a CFG for recognizing ontology objects from PODs a parsing strategy is needed. The XONTO system adopts a parsing technique based on the *Augmented Transition Networks* (ATNs) (see Section 2.3). In the XONTO-L system ATNs are enriched by object-oriented capabilities. Such structure is named *Object-Oriented Augmented Transition Network* (OO-ATN) and are constituted by a collection of transition networks that mutually call each other. In particular in OO-ATNs: (i) each transition network corresponds to a single nonterminal in the grammar and has a unique name; (ii) transition network arcs are labeled either by class names or object identified (i.e. nonterminal symbols) and hold functions and/or predicates; (iii) each transition network path, from the start state to the final state, corresponds to a rule for such a nonterminal, and the sequence of arc labels in the path is the sequence of symbols in the descriptor body (i.e. grammar rule).

XONTO parsing strategy has a RTN backbone. Naive RTN-parsers recursively attempts to find a path through the RTN starting from the axiom by using backtracking approaches [121]. So this kind of parsers could require exponential-time. More efficient parsing algorithms like Early or CYK [61, 56] are dynamic programming algorithms that can be implemented by inference systems with time complexity $O(n^3)$, where n is the number of symbols in the input (See Section 2.3.2).

For this reason the Descriptor Compiler produces RTNs based on logic programs of an inference system (i.e DLV+) that adopt a dynamic programming approach. Such an approach, by eliminating the repetitive solution of sub-problems inherently in backtracking techniques, reduces the problem of parsing a string by means of RTNs to polynomial-time. However an inference system iterates over all values of all free variables for each rule and for every instance of a rule of a logic program [57]. Thus, in order to optimize the parser, the Descriptor Compiler creates logic programs that avoid to apply unnecessary rules and iterate over free variables. More in detail, the compiler rewrite each grammar rule as a set of logic rules (one for each state of the transition network that represent the grammar rule). Thus, natural joins among chain of free variables are avoided. Furthermore final transition networks, produces by Descriptor Compiler, are deterministic, have the minimal set of nodes and transitions, and have no epsilon transitions.

Figure 12.2 depicts the RTN that encodes the descriptors number (6) that call the transition network related to the descriptor (7). In the figure transition networks related to nonterminals in descriptor (7) are not shown.

In the following the logic program that encodes the transition network `weatherTable` drawn in Figure 12.2 is shown.

```
weatherTable_finState_2d('v', X1, Y1, X2, Y2) :-
    weatherRecord_coord_2d(X, 'h', X1, Y1, X2, Y2).
weatherTable_finState_2d('v', X1, Y1, X2, Y3) :-
    weatherTable_finState_2d('v', X1, Y1, X2, Y2),
    weatherRecord_coord_2d(X, 'h', X1, Y2, X2, Y3).
```

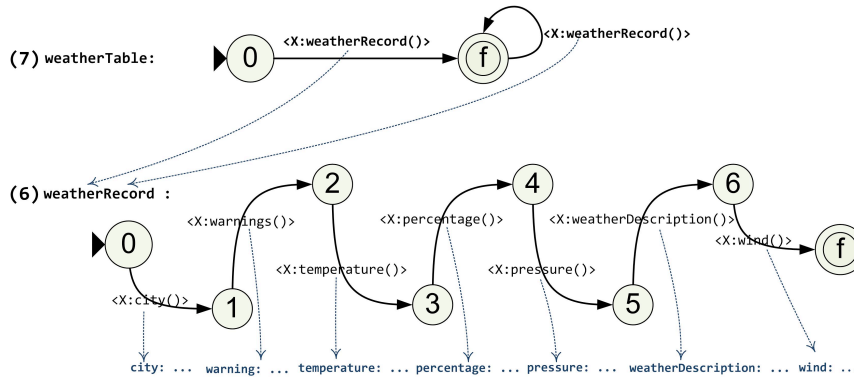


Fig. 12.2. A sample of some transition network aimed to recognize a weather table.

```

weatherTable_finState_2d('h', X1, Y1, X2, Y2) :-
    weatherRecord_coord_2d(X, 'v', X1, Y1, X2, Y2).
weatherTable_finState_2d('h', X1, Y1, X2, Y2) :-
    weatherTable_finState_2d('h', X1, Y1, X2, Y2),
    weatherRecord_coord_2d(X, 'v', X2, Y1, X3, Y2).

```

The previous program shows how coordinates of objects are handled during the parsing process. In fact, it enables to recognize both horizontal and vertical tables as required by grammar rules expressed by means of the descriptor (7). In particular, a `weatherTable_finState` is obtained by concatenating either exclusively horizontal or exclusively vertical `weatherRecord`. It is worthwhile noting that object coordinates are handled by using, for each object, an ad hoc relation that link the object with its coordinates. Schemas of such kind of relations are created by the Descriptor Compiler during compiling. Relations that hold coordinates are also arranged in taxonomies by following taxonomies of linked object. This way to represent and manage object coordinates allows to reduce the running time.

In order to fully compile descriptors the Descriptors Compiler enriches the RTNs by adding functions and predicates to transition networks arcs. Figure 12.3 depicts a fragment of the ATN obtained by compiling descriptor (7).

For instance, the transition network related to the descriptor number (7) (Figure 12.3) is translated by the Descriptors Compiler in the following logic program.

```

weatherTable_finState_2d(R, 'v', X1, Y1, X2, Y2) :-
    weatherRecord_coord_2d(X, 'h', X1, Y1, X2, Y2), R:=#addLast(X, []).
weatherTable_finState_2d(R1, 'v', X1, Y1, X2, Y2) :-
    weatherTable_finState_2d(R, 'v', X1, Y1, X2, Y2),
    weatherRecord_coord_2d(X, 'h', X1, Y2, X2, Y3), R1:=#addLast(X, R).

```

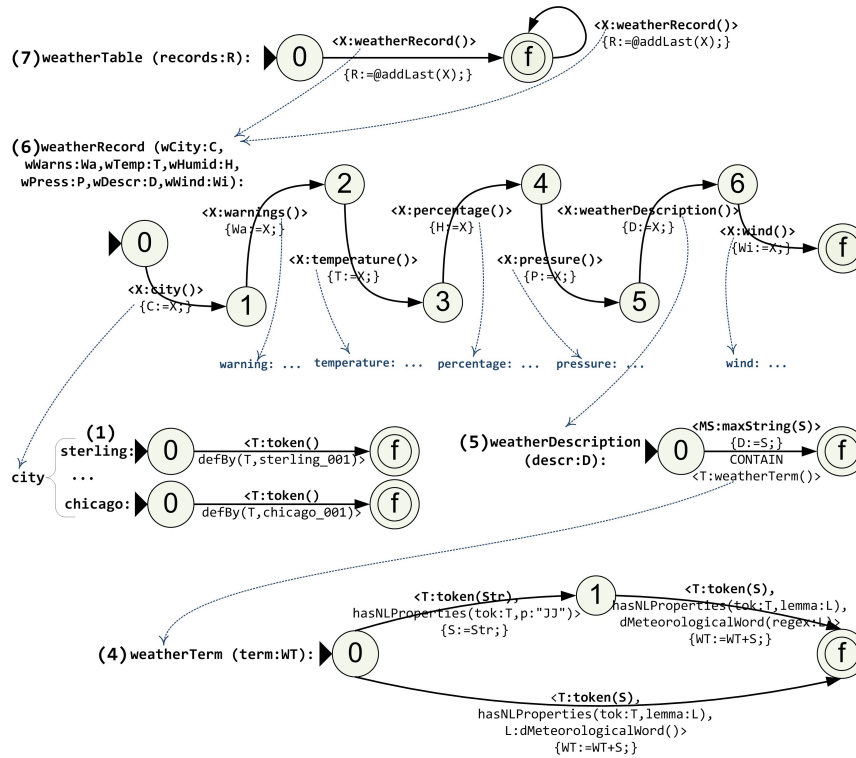


Fig. 12.3. Fragment of the OO-ATN aimed at recognizing and acquiring information about weather tables.

```

weatherTable_finState_2d(R, 'h', X1, Y1, X2, Y2) :-
    weatherRecord_coord_2d(X, 'v', X1, Y1, X2, Y2), R:=#addLast(X, []).
weatherTable_finState_2d(R1, 'h', X1, Y1, X2, Y2) :-
    weatherTable_finState_2d(R, 'h', X1, Y1, X2, Y2),
    weatherRecord_coord_2d(X, 'v', X2, Y1, X3, Y2), R1:=#addLast(X, R).

ID:weatherTable(R) :-
    weatherRecord_finState_2d(R, _, X1, Y1, X2, Y2),
    ID:=#getID("weatherTable", R).
weatherRecord_coord_2d(ID, DIR, X1, Y1, X2, Y2) :-
    weatherRecord_finState_2d(R, DIR, X1, Y1, X2, Y2), ID:weatherTable(R).
    
```

This program is the RTN previously presented enriched by functions that assign values to attributes. Functions are implemented by built-in (e.g. `R:=#addLast(X, [])`) executed at run-time outside of the logical program by invoking imperative external procedures. In the previous example when a table is recognized and final state verified the built-in `#getID("weatherTable", R)`

is executed in order to assign an (*oid*) to the new object. The (*oid*) is computed by considering class type and attribute values.

In Figure 12.3 the fragment of the ATN that allows to recognize objects of the class `city` (descriptor (1)) is also depicted. It is worthwhile noting that this example shows how semantic capabilities of the *SDO* approach are concretely exploited in the XONTO system. In fact, a call the transition network of the nonterminal `city` correspond to a call to all transition networks of the instance of the class `city`.

Rules Selector

The module Rule Selector takes in input user queries, the ontology and the OO-ATN obtained from the Descriptors Compiler and constructs the final logical program to execute by means of the Reasoning Engine. The module starts from the non-terminal symbols contained in the user queries (starting concepts), and explores semantic and logic dependencies among classes and objects in the ontology. For each class the set of logical programs implementing its class and objects descriptors are selected and gathered in a single logical program. The obtained program can be run by the Reasoning Engine for yielding the extraction of objects asked by means of user queries. For instance, the query `X:weatherTable([L])?` can be used to return weather tables contained in PDF document like that show in Figure 10.1.

Reasoning Engine

The *Reasoning engine* is constituted by DLV+ system (See Section 2.4.1 [158] which in turn is founded on the DLV system [109, 58].

The Reasoner takes in input the logical program created by the Rule Selector and the internal representation of the PODs created by the Document Analyzer. It provides the set of objects recognized within the document (both objects already existing in the SDO and extracted from the document itself) as asked in the user query. For example the previous query `X:weatherTable([L])?` returns the following set of instances:

```
@113:weatherTable(records:[...,@107,...]).
@107:weatherRecord(wCity:sterling,wWarns:@4ws,wTemp:@4tem,wHumid:@7hum,
    wPress:@6pr,wDescr:@7des,wWind:@6wind).
sterling:city(name:"Sterling",population:15451,inState:illinois).
@4ws:warnings(warns:["Tornado Watch",...]).
@4tem:temperature(value:86.7,unit:F).
@7hum:percentage(value:100).
@6pr:pressure(value:29.58, unit:in, trend:falling).
@7des:weatherDescription(descr:"Mostly cloudy").
@6wind(dir:S, value:8.0, unit:mph). ...
```


Instances returned in output can be serialized by using XML, RDF, etc. and used for many analytical and/or semantic-based applications. It is noteworthy that annotation can be performed automatically because coordinates associated to all objects extracted are known. Thus the XONTO system enables also documents annotation.

12.3 XONTO-G: Grammar-Based System for Extracting Objects

The grammar-based approach of XONTO (named XONTO-G) compiles a SDO in a pure grammatical production and perform a parsing strategy that exploits the logical 2-D representation of PODs and the SDO opportunely rewritten in terms of grammatical productions in order to yield the actual semantic and extracting information from PODs.

In this section the prototypical implementation of the XONTO-G system is described. Features and behavior of the prototypical implementation of the XONTO-G system are illustrated by means of a running example: the Web page shown in Figure 12.4¹, the Cartesian plane and MBRs are depicted in Figure 12.5. Moreover, the tractability of the grammatical approach is proven.

Figure 12.6 shows the prototype architecture. A SDO is used for many *ontology population processes* that are sequences of the following steps: preprocessing, 2-D matching, population. The input of a population process is constituted by a SDO produced by the *compiler*, an *unstructured document* and a *user query*. The output is the SDO+ that is the original SDO augmented by concept instances extracted from the input document. Compiling, 2-dimensional matching and population phases are described in the following. Whereas preprocessing phase is equivalent to the document analyzing accomplished by the XONTO-L system and already described in Section 12.2.

In the following the behavior of each system module is described in detail. For simplicity but without of generability, spatial production rules of descriptors will consider only **horizontal** and **vertical** direction of concatenation, and **containment**.

Compiler

The *compiler* translates a SDO in pure grammatical terms. Descriptors are extraction rules that constitutes an abstract way for expressing Δ productions extended by objects and 2-dimensional capabilities.

In order to formally define language the $\mathcal{L}(SDO)$, we extend $SG = \langle \Sigma, N, S, R_{card}, \Pi \rangle$ and $\Delta = \langle SG, Attr, Func, Pred \rangle$ by $SG_{\leq} = \langle \Sigma, N, S, R_{card}, \Pi_{\leq} \rangle$ and $\mathcal{AG}_{\leq} = \langle SG_{\leq}, Attr, Func_{\leq}, Pred \rangle$, respectively, making use of \mathcal{O}_z in such a way that: (i) $\Pi_{\leq} = \Pi \cup \{c_1 \rightarrow c_2 \mid c_1, c_2 \in C, c_2 \leq c_1\}$; (ii)

¹ <http://weather.yahoo.com>

Yahoo! My Yahoo! Mail Search: Web Search

YAHOO! NEWS Sign In
New User? Sign Up Weather Home - Help

Home U.S. Business World Entertainment Sports Tech Politics Science Health Travel Most Popular

Photos Opinion Local News Odd News Comics Weather Full Coverage Video/Audio Kevin Sites Site Index

Search: All News Advanced

The Weather Channel **Chicago Weather** Add Chicago Weather to Your My Yahoo! Page

[weather.com](#) [Change Location](#)

(About My Yahoo! and RSS - RSS Help)


Weather > North America > United States > Illinois > Chicago F° | C°

MORE FROM WEATHER.COM

Current conditions as of 7:51 am CDT

Partly Cloudy

Feels Like: 63°
Barometer: 29.97 in and rising
Humidity: 63%
Visibility: 10 mi
Dewpoint: 50°
Wind: 5.5 mph
Sunrise: 5:15 am
Sunset: 8:29 pm


63°
 High: 82° Low: 62°

[» Detailed Forecast](#)
[» Records & Averages](#)
[» Get Yahoo! Weather on your desktop](#)

Airport Conditions
Summer Vacation Rentals
Severe Alerts on your Desktop
Gardening Tips and Almanac
Local Golf Course Weather
Severe Weather Video

ADVERTISEMENT

TODAY	TOMORROW	SAT	SUN	MON	6-10 DAY
					Extended Forecast
Mostly Sunny	AM Showers	Scattered T-storms	Isolated T-storms	Sunny	
High: 79° Low: 59°	High: 82° Low: 62°	High: 79° Low: 60°	High: 77° Low: 58°	High: 78° Low: 61°	

Get Alerts: Mobile Email Snowfall Alerts Weather Bulletins

Fig. 12.4. Yahoo Chicago Weather Page

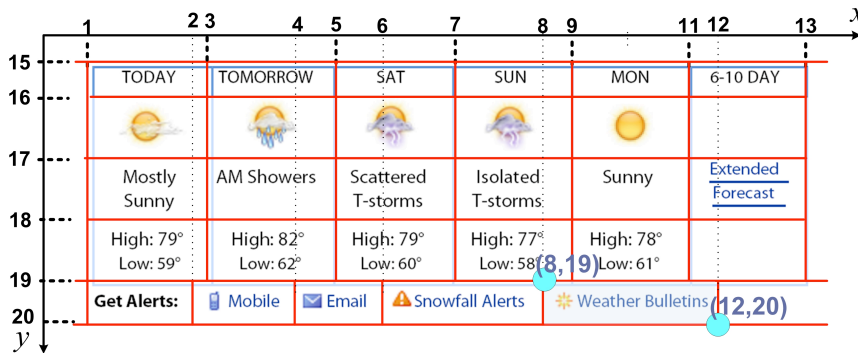


Fig. 12.5. Cartesian Plane and MBRs for the Table contained in Figure 12.4

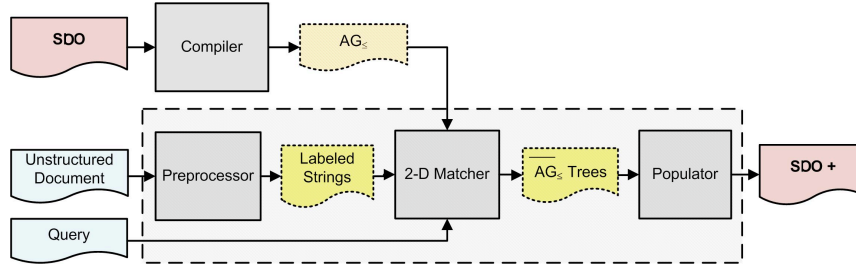


Fig. 12.6. Architecture of the SPO System Prototype.

$Func_{\leq}(p) = \{c_1.a := c_2.a \mid (a, t) \in \sigma(c_2) \text{ for some type } t\}$, for each production $p: c_1 \rightarrow c_2$ in $\Pi_{\leq} \setminus \Pi$.

Definition 12.2. Grammar \mathcal{AG}_{\leq} is equivalent to SDO , that is $\mathcal{L}(SDO) = \mathcal{L}(\mathcal{AG}_{\leq})$. ■

In the following examples, descriptors allow for recognizing and extract object of the class `weatherTerm`.

```
class weatherTerm(descr:string).
  sunny:weatherTerm("sunny").
  <sunny> -> <T:token(), hasLemma(tok:T, lemma:"sunny")>(12.1)
  shower:weatherTerm("showers").
  <shower>-> <T:token(), hasLemma(tok:T, lemma:"shower")>(12.2)
```

The descriptor of the class `weatherTerm` exploits linguistic capabilities. For instance, it allows to recognize the object `sunny` when in the text a token with lemma "sunny" is present.

The object descriptor number (12.1) is translated in term of \mathcal{AG}_{\leq} production rules as follows:

```
WEATHERTERM0 → TOK_SUNNY1,   id(0):=sunny,   value(0):"sunny",
                                     ##hasLemma(id(1), "sunny").
TOK_SUNNY0 → 'sunny', id(0):=#newID(),   value(0):"sunny"(12.3)
TOK_SUNNY0 → 'sunnier', id(0):=#newID(),   value(0):"sunnier".
TOK_SUNNY0 → 'sunniest', id(0):=#newID(),   value(0):"sunniest".
```

In this production the attributes `id` and `value` of the non terminal `WEATHERTERM` are set respectively to `sunny`(constant in \mathcal{O}^z) and "sunny" (string) by using the related functions when the predicate `##hasLemma(id(1), "sunny")` answers that the string recognized in the document has as lemma the

word `sunny`. This predicate is executed by the 2-D matcher because lemmas depend from the position of a word in a string.

The following descriptors allow to recognize and extract object of the classes `temperaturesPair` and `temperature`.

```
class temperaturesPair(high:integer, low:integer).
<temperaturesPair(high:H, low:L)> -> {integer TMP;}
  <temperature(value:T)> {TMP:=T;}
  <T:token(), defBy(T,rx_lowHigh)> (12.4)
  <temperature(value:T)> {H:=#max(TMP,T); L:=#min(TMP,T);}
  SEPBY <blankSequence>.
```

```
class temperature(value:integer).
<temperature(value:V)> ->
  <T:token(value:S), defBy(T,rx_d2)>{V:=#str2int(S);}12.5)
  <T:token(value:"o")>.
```

A weather `temperature` object is a sequence of a number (with two digits) and the symbol “°”. The function `#str2int` converts a string in the corresponding integer value. The objects recognized by means of the `temperature` descriptor, are exploited by the descriptor of the class `temperaturesPair` that describes a sequence of: a `temperature`, a `token` and a second `temperature` all separated by a `blankSequence` object. The construct `SEPBY` is “syntactic sugar” and expresses that each couple of objects must be *separated* by one or more blank characters. The higher and the lower temperatures are calculated in procedural mode.

The object descriptor number (12.5) is translated in term of production rules as follows:

```
TEMPERATUREPAIR0 → TEMPERATURE1 SEPARATOR2 TOK_RX_LOWHIGH3
SEPARATOR4 TEMPERATURE5,
  id(2) == blankSequence, id(4) == blankSequence
  high(0) := #max(value(1), value(5)),
  low(0) := #min(value(1), value(5)).
TOK_RX_LOWHIGH0 → ‘low’, id(0) := #newID(), value(0) := "low"(12.6)
TOK_RX_LOWHIGH0 → ‘high’, id(0) := #newID(), value(0) := "high".
TOK_RX_LOWHIGH0 → ‘min’, id(0) := #newID(), value(0) := "min".
TOK_RX_LOWHIGH0 → ‘max’, id(0) := #newID(), value(0) := "max".
```

This production is labeled as "horizontal" because it express an horizontal sequence of the non-terminals in left side. In particular, `TOK_RX_LOWHIGH` non terminal represents tokens defined by the regular expression translated in standard *AG* productions. Predicates $\#max(value_{(1)}, value_{(5)})$ and $\#min(value_{(1)}, value_{(5)})$ are evaluated by the populator because they do not depends from positions.

2-D Matcher

2-D matcher, shown in Figure 12.7, is composed by two main sub modules: *selector* and *parse tree builder*.

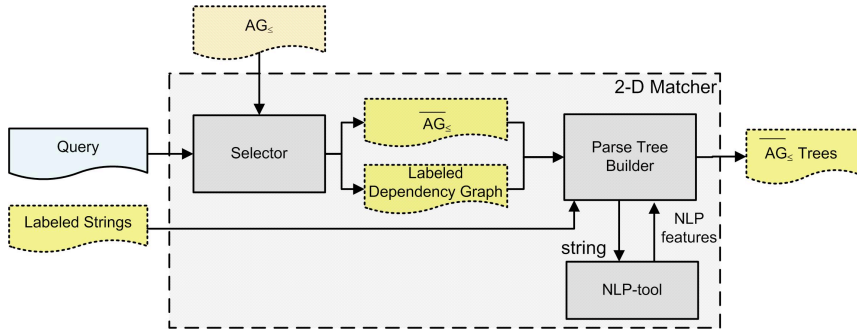


Fig. 12.7. Module 2-Dimensional Matcher of the SPO System Prototype.

Selector takes in input the user query and $\overline{AG_z}$. It starts from the non-terminal symbol contained in the user query (start concept), explores $\overline{AG_z}$ in order to identify the subset of productions $\overline{AG_z}$ that constitutes the grammar that has as axiom the start concept. Then the $\overline{AG_z}$ *labeled dependency graph* ($\overline{AG_z}^{LDG}$) is built up. As shown in Figure 12.8 for the weather forecast example, each node of the $\overline{AG_z}^{LDG}$ is equipped by a label that expresses which kind of spatial check must be executed.

Parse tree builder takes as input the document representation generated by the preprocessor, the $\overline{AG_z}$ and the $\overline{AG_z}^{LDG}$. The output of the *parse tree builder* are all the *admissible* parse tree of $\overline{AG_z}$ obtained by applying a suitable variant of the bottom-up version of Earley's chart parsing algorithm (See Section 2.3) that is able to handle strings contained in the 2-dimensional document representation. Figure 12.9 shows a sketch of the output $\overline{AG_z}$ parse tree corresponding to the encompassed area of the $\overline{AG_z}^{LDG}$ shown in Figure 12.8 when the user query is `X : weatherForecastTable([L])?`.

In XONTO-G we consider the following restrictions: The set *Func* contains *arithmetic expressions*, *string expressions*, *list expressions* and all the functions that allow to manipulate attribute values in P-TIME. Furthermore,

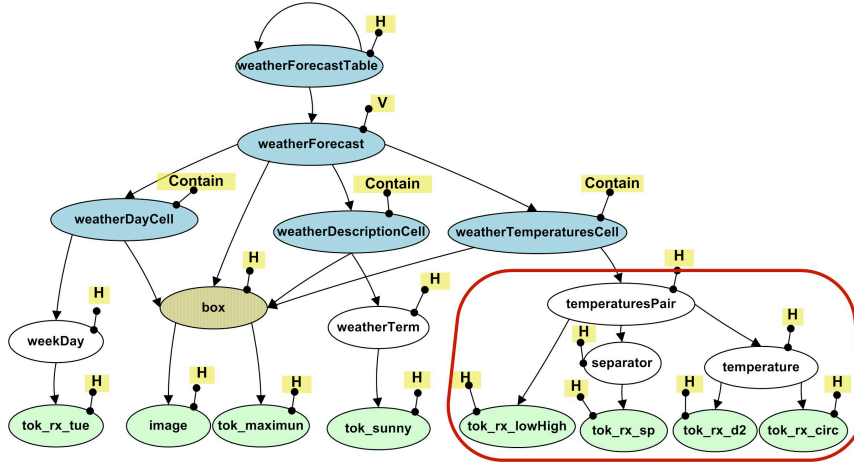


Fig. 12.8. Labeled Dependency Graph

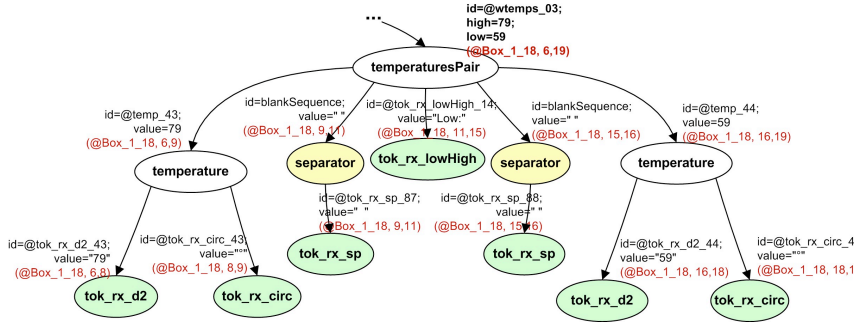


Fig. 12.9. A sketch of the parse tree resulting from the query $X:weatherForecastTable()$?

let p be any production in SDO, then each attribute in the right-hand side of p appears *at most once* in all the expressions of $Func(p)$. In the following the polynomial complexity of the approach is proven.

Lemma 12.3. Let t be any parse tree² of \mathcal{AG}_{\leq} and $x(t)$ the string that yields t such that $\|x(t)\| = n$. Then, each attribute value in t has length $\mathcal{O}(n)$.

Proof. Let x_1, \dots, x_k be the attributes of \mathcal{AG}_{\leq} . Without loss of generality, we assume to deal with attributes on strings. Given a node v in t , we denote by $\|v\|$ the length of the string obtained concatenating all the attribute values in v . Since any attribute can be exploited at most once in all the functions of a production, then $\|v\| \leq c + |x_1^1| + \dots + |x_k^1| + \dots + c + |x_1^h| + \dots + |x_k^h|$ holds, where

² Note that t could also be invalid for \mathcal{AG}_{\leq} .

any x_i^j is attribute x_i in the j^{th} child of v ($0 \leq i \leq k$), and c is the length of the longest (constant) string in $Func_{\mathcal{G}_{\leq}}$. So the rise in length of each node is at most $k * c$ and $size(t) \leq |N| * (2n - 1)$ where $|N|$ is the number of \mathcal{AG} non-terminal symbols, then $\|\rho(t)\| \leq k * c * |N| * (2n - 1)$ where $\rho(t)$ is the root of t . Clearly any attribute value in t has length $\mathcal{O}(n)$. ■

Definition 12.4. Given a SDO in which \mathcal{AG}_{\leq} is unambiguous, then problem OG-PARSE is defined as follows. Given an input string w , does w belong to $\mathcal{L}(\mathcal{AG}_{\leq})$? ■

Theorem 12.5. SDO-PARSE is **P**-complete.

Proof. (Membership) Let w to parse for membership in $\mathcal{L}(\mathcal{AG}_{\leq})$. Since \mathcal{G}_{\leq} is unambiguous, so if $w \in \mathcal{L}(\mathcal{G}_{\leq})$, there exists exactly one parse tree t for \mathcal{G}_{\leq} . Then, all the functions in $Func_{\leq}$ related to the nodes of t can be computed from the leaves of t to its root. The union of \mathcal{O}_{\leq} with the new objects generated in t composes \mathcal{O}_{\leq}^+ . Hence, for each non-leaf node v_0 of t , having children v_1, \dots, v_h , are evaluated the related predicates. In particular, to each query $\langle p(X_1, \dots, X_n), \mathcal{Q} \rangle$ in v_0 corresponds the evaluation of expression $\mathcal{C}(\mathcal{O}_{\leq}^+) \cup \mathcal{Q} \models p(z_1, \dots, z_n)$, where z_1, \dots, z_n are attribute values of node v_i ($0 \leq i \leq h$). Consider now that t is generated in polynomial time, and all functions are polynomial time computable. Moreover, let $\mathcal{D} = \mathcal{C}(\mathcal{O}_{\leq}^+) \setminus \mathcal{C}(\mathcal{O}_{\leq})$ the ground facts representing the new objects, then \mathcal{D} is polynomial in the size of w as shown in the proof of by Lemma 1. Therefore, since deciding $\mathcal{Q} \cup \mathcal{D} \models p(z_1, \dots, z_n)$ is **P-complete** [45] when Datalog query \mathcal{Q} is fixed, whereas database \mathcal{D} and atom $p(z_1, \dots, z_n)$ are an input (data complexity), then $\mathcal{C}(\mathcal{O}_{\leq}) \cup \mathcal{D} \cup \mathcal{Q} \models p(z_1, \dots, z_n)$ is polynomial in $\|w\|$ because $\mathcal{C}(\mathcal{O}_{\leq})$ and \mathcal{Q} are fixed while $p(z_1, \dots, z_n)$ and \mathcal{D} are an input.

(Hardness) It is enough to notice that since deciding $\mathcal{Q} \cup \mathcal{D} \models p(z_1, \dots, z_n)$ is **P-complete**, so $\mathcal{C}(\mathcal{O}_{\leq}) \cup \mathcal{D} \cup \mathcal{Q} \models p(z_1, \dots, z_n)$ is **P-hard**, then OG-PARSE cannot be easier. ■

Populator

The populator, takes all the admissible parse trees as input. For each of them it evaluates functions that assign values to object attributes (including OIDs) and predicates that are queries on the SDO that check if obtained objects are allowed for SDO. If the check goes well, obtained objects are added to the initial SDO in order to produce the SDO+. It is noteworthy that if new objects are added to SDO then a subset of the strings in input constitutes a language for $L(\mathcal{AG}_{\leq})$. By considering the user query $X : \text{weatherForecastTable}([L])?$ new objects added to the initial SDO are shown in the following.

```
@t_001:weatherForecastTable(weathers:[@wf_01,@wf_02,@wf_03,@wf_04,@wf_05]).
@wf_01:weatherForecast(day:@wday_03,descr:@tok_max09_01,hTemp:79,lTemp:59).
...
@wf_05:weatherForecast(day:@wday_07,descr:@tok_max23_01,hTemp:78,lTemp:61).
```

```
@wday_03:weekDay(day:"Thursday").  
@tok_max_09_01:tok_maximun(value:"Mostly Sunny").  
...  
@wday_07:weekDay(day="Monday").  
@tok_max_23_01:tok_maximun(value:"Sunny").
```

12.4 Discussion

In this section the ontology-based system for semantic IE from PDF documents, named XONTO, has been presented. The Self-Describing Ontologies paradigm (SDO), which the system is founded on, has also been described by means of a running example, and its tractability has been proved. The paradigm brings together: the expressive power of ontologies, the Spatial Grammar with attribute formalisms, and the Spatial document model. This approach allow for: exploiting semantics represented in a Knowledge Base in ordert to recognize information organized in both textual and tabular form, and directly storing extracted information in a Knowledge Base as classes instances (i.e. objects). So, the most interesting feature of XONTO system is that it exploits the knowledge represented in an ontology for extracting information and populate the ontology itself with objects extracted from PDF documents. XONTO system enables to turn in structured form unstructured information. This way extracted information can be queried and analyzed by means of already existing techniques coming from the database world. Features of the XONTO system empower unstructured information management capabilities of existing applications for creating valuable solutions for enterprises.

Visual Information Extraction

This chapter aims at showing how novel approaches, models and algorithms described in previous chapters have been implemented in order to provide to final users better visual capabilities in extracting information from presentation-oriented documents. In the following will be first shown the XPath and SILA systems visual features that enable to query and extract information from Deep Web sites by using visual capabilities. Then functionalities for wrapping information and tables from PDF documents of the PDF-TREX system will be explored.¹

13.1 Visual Features in the SILA and XPath System

The visual interface of the XPath and SILA system is shown in Figures 13.1 and 13.2. They embeds the Mozilla Firefox browser² and includes several windows, such as: (i) the browser window that renders web pages and enables to both visually select the area of a Web page for which the user intend to define a wrapper and visually show wrapper obtained by induction; (ii) a DOM tree window that shows the HTML DOM tree of the current web page rendered by the browser; (iii) a navigation window that allows the user to inspect and manipulate navigation and extraction actions needed for a complete information extraction task. A mapping window can be open on demand by the user for inspecting the structure wrappers and defining mappings of extracted records to specific XML structures. (iv) a query windows (see Figure 13.1) that allows for writing XPath queries and shows the textual results (visual results can be highlighted on the document in the browser window as well as saved in other document format, such as XHTML and XML).

¹ Features shown in this section are used in the context of the spin-off company Altilia srl (www.italiagroup.com) of the Italian National Research Council that builds semantic technologies in the field of content management and text analytics.

² <http://www.mozilla.org/>

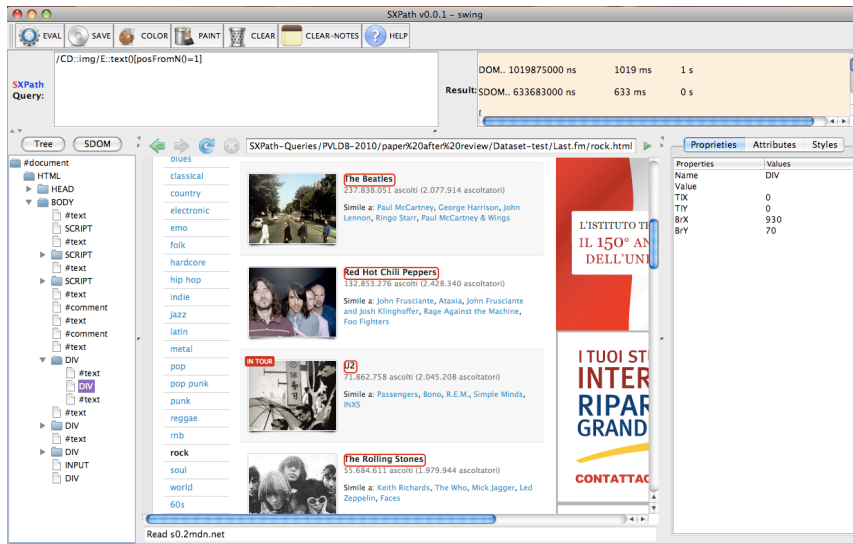


Fig. 13.1. Visual Interface of the SXPath System

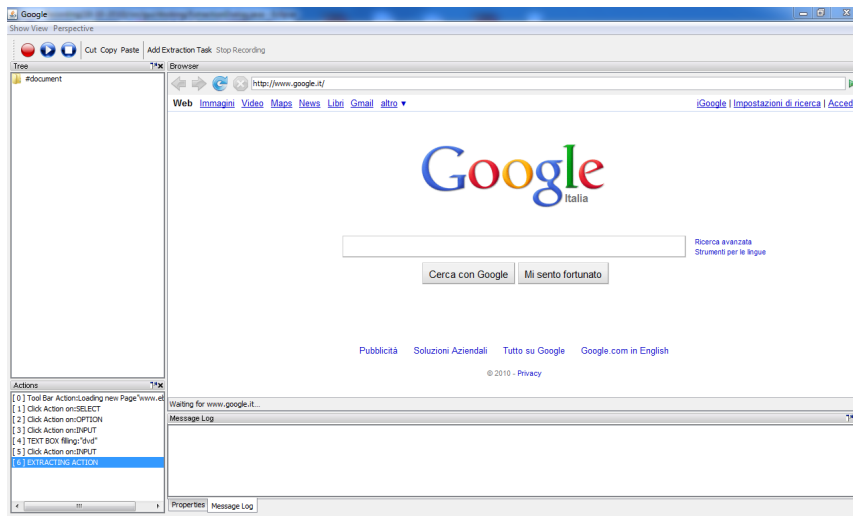


Fig. 13.2. Visual Interface of the SILA System

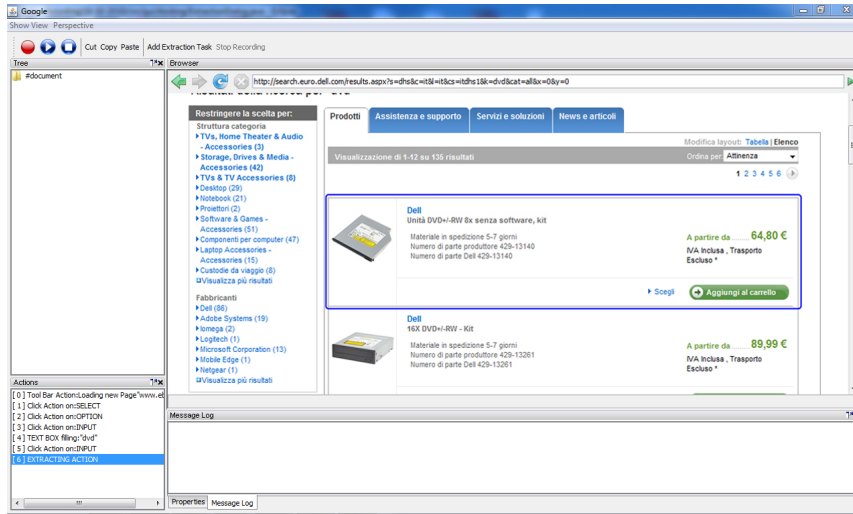


Fig. 13.3. Visual Selection of a Data Record for Wrapper Learning

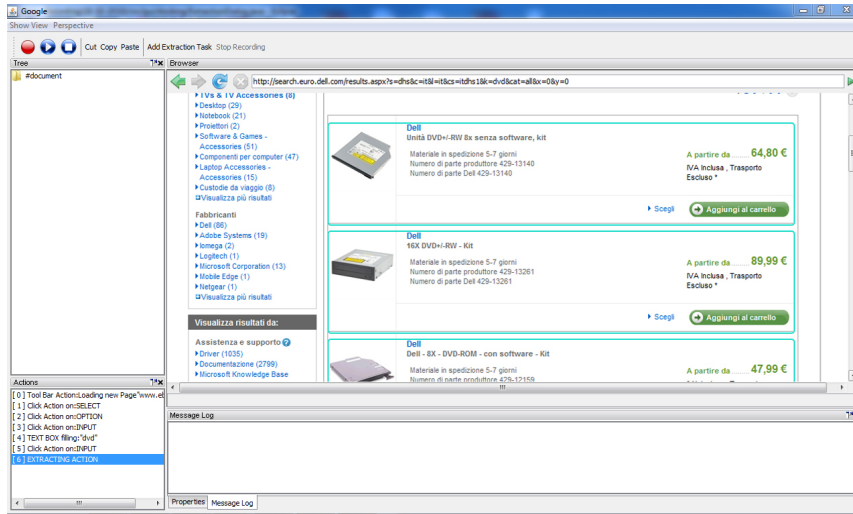


Fig. 13.4. Data Records Identified by the Visual Wrapper

In Figure 13.3 is shown how users can select a portion of a page for which they need to define a wrapper. By using only the mouse they can select an object that intend to extract. The system automatically selects the portion of the SDOM related to the visualized area and defines the wrapper. Wrappers can be composed either as: (i) a set of SXPath rules that allow for identifying the portion of the page by using its visual structure; or as (ii) a set of productions of the SGG describing the spatial arrangement of elements that the user intent to extract by the wrapper. Navigation actions needed for reaching a specific Web page, filling Web form, and obtaining desired pages or lists of data records on multiple pages, can be visually registered navigating the Web by the embedded browser. An ad hoc component of the system is aimed at registering user actions and made them repeatable. Extraction rules for pages containing multiple data records can be automatically learned by the SILA algorithm and stores as extraction actions. In this case the system visually show to users the set of records that can be extracted by using learned extraction rules. Navigation and extraction actions can be edited, modified and stored by users. Furthermore actions can be automatically executed in order to perform navigation and extraction tasks in batch mode.

Visual navigation and extraction features are very powerful and allow users to write no code for defining complex extraction tasks.

13.2 Visual Features in the PDF-TREX System

The visual interface of the PDF-TREX system is shown in Figure 13.5, it embeds the PDFBox PDF rendering library³, and the Open Office spreadsheet⁴. The interface includes a main menu, a tool bar, and some windows, such as: (i) the explorer window that allow for browsing the file system in order to find PDF document to process; (ii) the PDF rendering window that visualizes opened PDF documents. This window is interactive because it show to users recognized tables and text areas by graphical grids. Furthermore users can modify the result of an automatic table extraction task simply by using mouse clicks on the visualized grids. So users can merge tables, remove table rows and columns, add table by specifying to the system the area in which a table must be found, etc. It is worthwhile nothing that by using this windows users can specify extraction template (wrappers) to use multiple times on multiple documents. Users have just to draw rectangles by using mouse clicks. This way the wrapping design process is completely visual. Moreover, the system is able to suggest a possible wrapper by using a wrapper induction approach for PDF document. Wrappers suggested by the system can be edited and modified by users using only mouse clicks; (iv) the spreadsheet window that visualizes tables recognized and extracted on PDF documents.

³ <http://pdfbox.apache.org/>

⁴ <http://www.openoffice.org/>

For each document opened in the PDF rendering window there is a tab in the spreadsheet that contain recognized tables. Extracted tables can be saved in HTML, XLS, XML, CSV format.

Main menu and tool bar allow for performing system actions on visualized PDF documents. Users can automatically recognize tables and text areas contained in a multipage document, learn a wrapper for a give type of documents, visually design wrappers for specific documents, edit recognized table and text areas in order to remove columns, rows, text areas, add table and text areas, merge tables and text areas, etc. The same set of feature can be also called by using the system contextual menu attached to right click or the main menu available on the menu bar.

In the following an example of table extraction is shown. In Figure 13.6 is depicted a PDF document opened in the PDF-TREX system. By selecting the *extract table* menu item the system automatically recognize tables and text areas in the document and visually shows to the user the result by using a graphical grid as shown in Figure 13.7. Extracted tables are also available in the embedded spreadsheet in structured form.

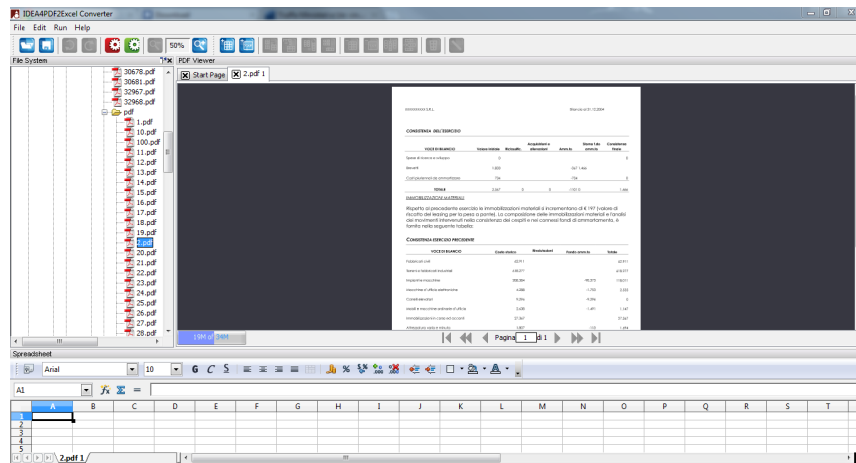


Fig. 13.5. Teh User Interface of the PDF-TREX System

A user interested in the extraction of a particular table, can use the *add table* menu item. So s/he can select, by using only the mouse, the area of the document containing the table and obtain the resulting table highlighted in the document and structured in the spreadsheet as shown in Figure 13.7. Tables can be stored in HTML, XLS, CSV, XML form.

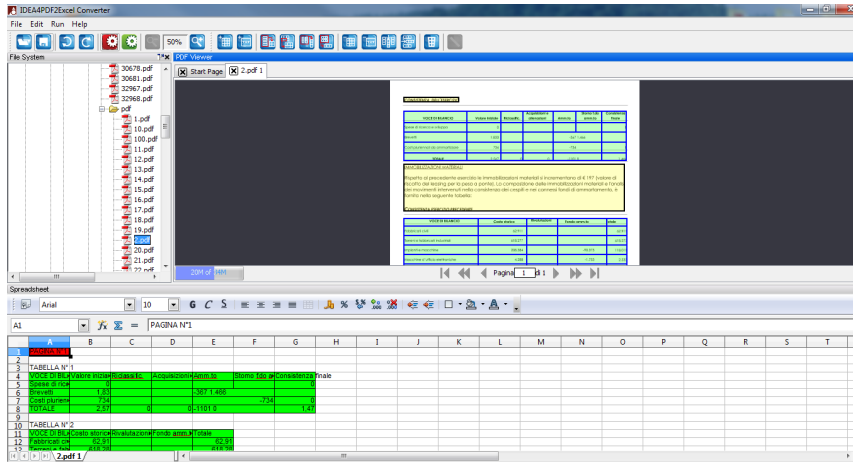


Fig. 13.6. Tables Automatically Recognized In a PDF Document

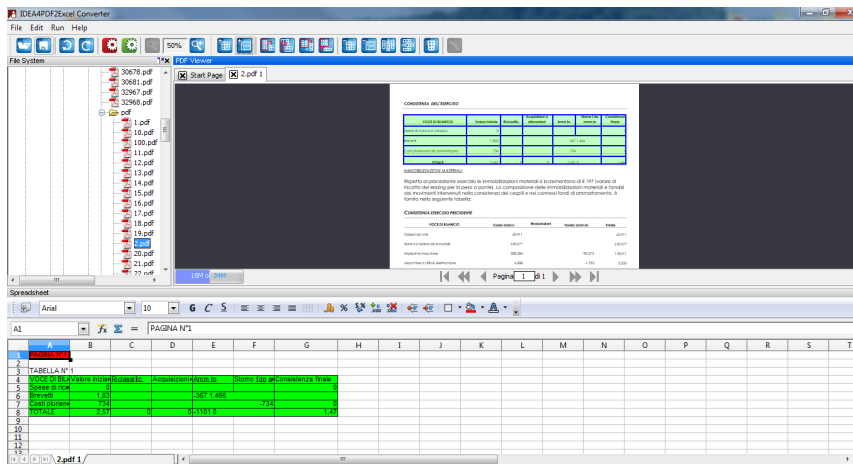


Fig. 13.7. A Table Recognized by a Visual Selection

Real World Applications

This chapter aims at showing how approaches, methods and algorithms described in this PhD thesis can help in real life application scenarios. The unified spatial document model, as well as automatic information extraction techniques and query languages, in fact, support the definition of visual tools capable to simplify user activities in managing contents of presentation-oriented documents and getting knowledge from them. In the following of this section is shown how tools and approaches described in this thesis are used for managing: clinical processes [144, 141], company records like invoices, and official bulletin of a public administration.

14.1 Medical Process Management

In this section is described an Ontology-based Clinical Knowledge Representation Framework (OCKRF), that aims at supporting a semantic process-centered vision of health care practices, and its prototypical implementation. The OCKRF allows for representing and managing both static and dynamic aspects of medical knowledge. It allows for extracting information from clinical records, and enables to adopt semantics for better designing, executing, controlling and managing clinical processes. In the following are shown: (i) the process meta-model adopted in the OCKRF also by using a running example; (ii) theoretical foundations of OCKRF and how to use it by example; (iii) an example of information extraction from medical records.

14.1.1 Process Modelling

A significant amount of research has been already done in the specification of mechanisms for process modeling (see, [73] for an overview of different proposals). The most widely adopted formalism is the control flow graph, in which a workflow is represented by a labeled directed graph whose nodes correspond to the activities to be performed, and whose arcs describe the

precedences among them. In this paper, we adopt the graph-oriented workflow meta-model shown in Figure 14.1.a and 14.1.b, inspired by the JPDL [98] process modeling approach. The adopted meta-model: (i) covers the most important and typical constructs required in workflow specification; (ii) allows to implement the OCKRF by using the JBPM workflow engine; (iii) allows to use workflow mining techniques grounded on graph-oriented meta-models. Since our scope is to extend the meta-model by semantic features we need firstly to formally define it as the following 6-tuple:

$$\mathcal{P} = \langle N, A_r, E_v, A_n, T_k, E \rangle$$

where:

- N is a finite set of *nodes* partitioned in the following subsets: task nodes N_T (that represent activities in which a human perform tasks), subprocess nodes N_{SP} (that model activities that refer processes external to the current one), group nodes N_G (that represent a set of nodes that can be executed without a specific order), custom nodes N_C (that model activities in which custom methods can be executed and handled automatically), wait nodes N_W (that represent activities that temporary stop the execution while they execute methods), join nodes N_J , fork nodes N_F (that are respectively used to combine or split execution paths) and decision nodes N_D (that allow to control the execution flow on the base of conditions, variables or choices performed by human actors).
- A_r is a set of *actors*. Actors can be human or automatic. They represent the agents that execute a given task or activity.
- A_n is a set of *actions*. An action is a special activity that can be performed as answer to the occurrence of an event.
- T_k is a set of *tasks* that represent tasks to execute in task nodes.
- $E = \{\langle x, y \rangle : x \in N_{From} \wedge y \in N_{To}\}$ is a set of *transitions* in which the following restrictions hold, when $N_{From} \equiv N_{FN} \cup N_D$ then $N_{To} \equiv N_{FN} \cup N_{FCN} \cup N_{end}$ and when $N_{From} \equiv N_{FCN}$ then $N_{To} \equiv N_{FN} \cup N_D$. Moreover, for each process there is a transition of the form $e_{start} = \langle N_{start}, y \rangle$ where $y \in N_{FN}$ and one of the form $e_{end} = \langle x, N_{end} \rangle$ where $x \in \{N_{FN} \cup N_D\}$. The subset $E_d \subset E$ where $E_d = \{\langle x, y \rangle : x \in N_D \wedge y \in N_{FN}\}$ is the set of *decisions*. A decision relates a decision node to a flow node and could hold a *decision rule* that is used at run-time to automatically control the execution flow of a process.
- E_v is a set of *events*. An event causes the execution of an action that constitutes the answer to the event. An event can be, for example, the throwing of an exception during the execution of a task.

A Clinical Process for Caring Breast Neoplasm

This section describes a clinical process for caring the breast neoplasm (Figure 14.1.c). This process will be used in the rest of the paper as running example

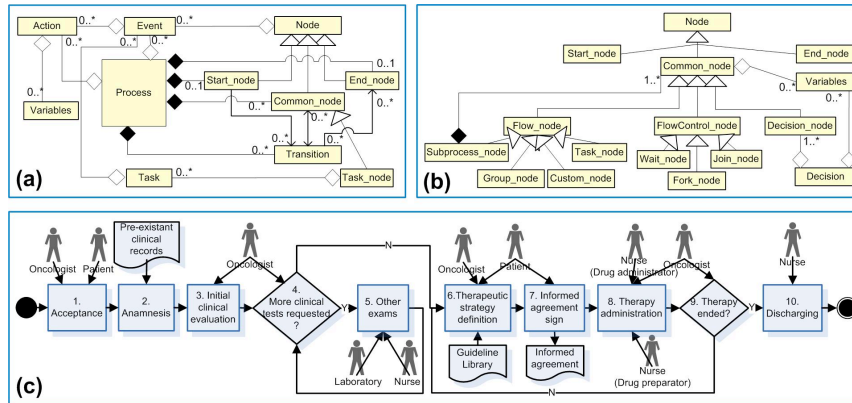


Fig. 14.1. (a) The process meta-model. (b) The nodes hierarchy. (c) A clinical process for caring the breast neoplasm

for demonstrating OCKRF features. The example considers practices carried out in the oncological ward of an Italian hospital, hence it is not general but specific for the domain of the considered ward. The clinical process is organized in the following 10 activities:

1. Task node *Acceptance* models patient enrollment. A patient arrives to the ward with an already existing clinical diagnosis of a breast neoplasm. This activity can be performed manually by an oncologist that collects patient personal data, and directly acquiring information from medical records (Electronic Medical Records EMRs). Such information are stored into an Ontology. The extraction is performed exploiting the semantic information extraction approach described in Chapter 12.
2. Group node *Anamnesis* represents a set of anamnesis activities: *general anamnesis* in which physiological general data (e.g. allergies, intolerances) are being collected; *remote pathological anamnesis*, concerning past pathologies; *recent pathological anamnesis*, in which each data or result derived from examinations concerning the current pathology (or pathologies) are acquired. These activities can be executed without a specific order, and exploiting semantic extraction rules that enable the recognition of information into unstructured source like EMR.
3. Task node *Initial clinical evaluation* allows to acquire the result of an examination of the patient by an oncologist.
4. Decision node *More clinical tests requested* represents the decision to perform or not additional examination on the patient.
5. Group node *Other exams* models possible additional clinical tests. If requested these tests are conducted to find out general or particular conditions of patient and disease not fully deducible from the test results already available.

6. Task node *Therapeutic strategy definition* models the selection of a guideline with related drug prescription. At execution time the physician picks a guideline (selected among the guidelines already available in the knowledge base) that depends upon actual pathology state as well as other collected patient data.
7. Task node *Informed agreement sign* models the agreement of the patient concerning understanding and acceptance of consequences (either side effects or benefits) which may derive from the chosen chemotherapy, and privacy agreements.
8. Sub-process *Therapy administration*, models a subprocess that constitutes the guideline to execute for caring the patient.
9. Decision node *Therapy ended* models a decision activity about effects of the therapy and the possibility to stop or continue cares.
10. Task node *Discharging* models the discharging of the patient from the ward end allows to acquire final clinical parameter values.

In the activities (6) and (8) risk and error conditions can be identified. At each guideline, chosen in (6), corresponds a prescription of drugs (chemotherapy). Hence the computation of doses, which may depend on patient's biomedical parameters such as body's weight or skin's surface, is required. Cross-checking doses is fundamental here, because if a wrong dose is given to the patient the outcome could be lethal. Furthermore, therapy administration ((8)-th activity) must contain checks that aims at verify type and quantity of chemotherapeutic drugs to submit to the cared patient.

14.1.2 Ontology-based Clinical KR Framework

The Ontology-based Clinical Knowledge Representation Framework (OCKRF) is based on the DLP+ ontology representation language [158] that, beside complete and expressive ontology representation features, holds also powerful ASP reasoning capabilities [58, 109] over represented knowledge. Thus the OCKRF allows to represent in a combined way clinical processes (procedural medical knowledge) and medical ontologies (declarative medical knowledge). By means of user friendly interfaces the framework is well suited for enabling agile knowledge representation. Furthermore, it provides powerful reasoning mechanisms that works over represented knowledge. OCKRF reasoning capabilities allow to express decisions, risk and error rules in a declarative way. More in detail, the OCKRF allows to represent extensional and intensional aspects of both declarative and procedural medical knowledge by means of: (i) *Medical Ontology and Clinical Process Schemas*. The former expresses concepts related to different medical domains (e.g. diseases, drugs, medical examinations, medical treatments, laboratory terms, anatomy, patients administration, risks). Ontology contents can be obtained by importing other existing medical ontologies and thesaurus or by means of direct manual definition. The latter are expressed according with the workflow meta-model illustrated in Section

14.1.1. The key idea which the framework is based on is that elements of the workflow meta-model (i.e. processes, nodes, tasks, events, transitions, actions, decisions) are expressed as ontology classes. This way workflow elements and medical knowledge can be easily combined in order to organize clinical processes and their elements as an ontology; (ii) *Ontology and Process Instances* both expressed in term of ontology instances. In particular, ontology class instances can be obtained by importing them from already existing medical ontologies or by creating them during process execution. Clinical process instances are created exclusively during process execution. Instances are stored in a knowledge base; (iii) *Reasoning Tasks* that express decision and risk rules computed by exploiting reasoning capabilities of DLP+ [158, 109]. More formally an ontology in the OCKRF is the 5-tuple: $\mathcal{O} = \langle D, A, C, R, I \rangle$. In the following the meaning of \mathcal{O} and the DLP+ language syntax (that express them) are explained by describing the implementation of the running example presented in Section 14.1.1.

Medical Ontology and Clinical Process Schemas

Schemas are expressed by using elements of D , A , C and R in \mathcal{O}_z that are *finite* and *disjoint* sets of entity names respectively called *data-types*, *attribute-names*, *classes* and *relations*. The set of classes C is organized in taxonomies and partitioned in two subsets: (i) the set of process classes $C_P = N \cup A_r \cup A_n \cup T_k \cup E_v$ that represents elements of the workflow meta-model. It is constituted by the union of classes representing nodes, actors, actions, tasks and events; (ii) the set of medical classes C_M that represent concepts related to different medical domains. The set R is partitioned in two subsets: (i) the set of relations $R_P = E \cup D_c$ aimed at representing transitions and decisions; (ii) the set of relations R_M used for representing medical ontologies.

A class can be thought as an aggregation of individuals (objects) that have the same set of properties (attributes). From a syntactical point of view, a class is a name and an ordered list of attributes identifying the properties of its instances. Each attribute is identified by a name and has a type specified as a data-type or class. In the following the DLP+ implementation of the workflow meta-model is presented. In particular, *nodes* in C_P are implemented by using the class hierarchy (built up by using `isa` key-word) shown below.

```
class process(name:string).
class node(name:string, container: process, start_time:integer, end_time:integer).
  class start_node() isa{node}.
  class end_node() isa{node}.
  class common_node () isa{node}.
    class flowControl_node() isa{common_node}.
      class fork() isa{flowControl_node}.
      class join() isa{flowControl_node}.
      class wait_node() isa{flowControl_node}.
    class flow_node() isa{common_node}.
      class task_node(tasks:[task], handler:human_actor) isa{flow_node}.
      class custom_node(handler: automatic_actor, method:string) isa{flow_node}.
      class group_node(nodes:[node]) isa{flow_node}.
      class sub_process_node(sub_proc: process) isa{flow_node}.
```

```

class decision_node(handler:actor) isa{common_node}.
  class automatic_decision_node(handler:automatic_actor) isa{decision_node}.
  class manual_decision_node(task:task, handler:human_actor) isa{decision_node}.

```

Task nodes and manual decision nodes contain *tasks* that are performed by humans. Tasks (`class task(name: string).`) collects values of activity variables given in input by human actor. *Actors* of a process (that can be human or automatic) represent the agents that execute a given task. They are represented by means of the following classes in C_P :

```

class actor(name:string).
  class human_actor() isa {actor}.
  class automatic_actor(uri:string) isa {actor}.

```

During the process enactment, by running risk and errors rules, *events* may occur. Furthermore, an event can be generated by an exception during the execution of a task. Events, and related actions to performs in response, are represented in C_P by the following classes.

```

class event(relativeTo:object, timestamp:integer).
  class node_event(relativeTo:node) isa{event}.
  class task_event(relativeTo:task) isa{event}.
  class process_event(relativeTo:process) isa{event}.
class action(method:string).

```

Relationships among objects are represented by means of relations, which like classes, are defined by a name and a list of attributes. *Transitions* and *decisions*, in R_P , that relate couple of nodes, are represented by means of the following ontology relations.

```

relation transition(name:string, from:node, to:node).
relation decision(name:string, from:decision_node, to:node).

```

When the user defines a specific process schema s/he can specialize original meta-model elements for adding new semantic attribute required by the specific process. In the following are shown some classes representing nodes of the running example depicted in section 14.1.1.

```

class acceptance_node(tasks:[acceptance_form],handler:physician) isa{task_node}.
class anamnesis_node(nodes:[general_anamnesis_node,remotePathological_anamnesis_node,
  recentPathological_anamnesis_node]) isa {group_node}.
class recentPathological_anamnesis_node(tasks:[pathology_form],handler:physician)
  isa {task_node}.
class therapeutic_strategy_definition_node(tasks:[therapeutic_strategy_form],handler:nurse)
  isa {task_node}.
class therapy_administration_node(sub_process:therapy_administration_process)
  isa{sub_process_node}.
class more_tests_node(task:more_tests_form) isa{manual_decision_node}.

```

`acceptance` and `therapeutic_strategy_definition` process activities are represented as subclasses of `task_node` class, in fact they represent activities in which tasks consist in the execution of forms filled by humans. Whereas `anamnesis_node`, which `Recent Pathological anamnesis` activity belongs to, is represented as a subclass of `group_node` class. `therapy_administration_node` and `more_tests_node` are specializations of `sub_proc_node` and `decision`

_node respectively. Human actors that operate in the clinical process used as running examples could be physicians, nurses and patients. They are represented by a person hierarchy that exploits multiple inheritance capabilities of DLP+ in order to express that persons are also human actors of the clinical process.

```
class person(fiscalCode:string,name:string,surname:string,sex:sex_type,
             bornDate:date,address:address).
class patient(hospitalCard:string, weight:float, heighthCm:float) isa {person,human_actor}.
class healthCareEmploy(occupation:string, role:string) isa {person,human_actor}.
class nurse() isa {healthCareEmploy}.
class physician() isa {healthCareEmploy}.
```

Class schemas representing tasks related to task-nodes can be expressed by using the following class schemas. Attribute types can be classes represented in C_M expressing different medical concepts (e.g. diseases, drugs, body parts). During task execution values of resulting class instances are obtained from fields filled in forms.

```
class task(name: string).
class acceptance_form(patient:patient, acc_date:date) isa{task}.
class pathology_form(disease:disease) isa{task}.
class chemotherapeutic_strategy_form(strategy:therapeuticStrategy) isa{task}.
class more_tests_form(choice:boolean)isa{task}.
```

In a clinical process, an event can be activated by an exception during the execution of a node or by a reasoning task aimed at control possible risks and errors. A reasoning task checks parameters values of running node and already acquired node instances and throws an event related to an error. An example of different kinds of possible errors is shown in the following taxonomy, where the attribute `msg` of the class `view_msg` (action) is the message to display when the error occurs.

```
class task_event(relativeTo:task) isa{event}.
class medicalError(msg:string) isa{task_event}.
class drugPrescriptionError() isa {medicalError}.
class view_msg(msg:string) isa {action}.
```

Class schemas in C_M expressing knowledge concerning anatomy, breast neoplasm disease and related therapies and drugs have been obtained (imported) from the Medical Subject Headings (Mesh) Tree Structures, the International Classification of Diseases (ICD10-CM).and the Anatomical Therapeutic Chemical (ATC/DDD).

```
class anatomy(name:string).
class bodyRegion() isa {anatomy}.
class disease(descr:string).
class neoplasm() isa {disease}.
class malignant_neoplasm() isa {neoplasm}.
class primarySited_neoplasm(site:bodyRegion,zone:string) isa {malignantNeoplasm}.
class breast_primarySited_neoplasm() isa {primarySited_neoplasm}.
class drug(name:string, ddd:float, unit:unitOfMeasure,admRoute:[string], notes:string).
class antineoplasticAndImmunomodulatingAgent() isa {drug}.
class endocrineTherapy() isa {antineoplasticAndImmunomodulatingAgent}.
class hormoneAntagonistsAndRelatedAgents() isa {endocrineTherapy}.
class enzymeInhibitors() isa {hormoneAntagonistsAndRelatedAgents}.
class hormoneAndRelatedAgents() isa {endocrineTherapy}.
```

```

        class estrogens() isa {hormoneAndRelatedAgents}.
class code(c:string).
    class icd10Code(chapter:integer, block:string,category:string, subCat:string) isa {code}.
    class mesh08Code(category:string, subCat:string) isa {code}.
class therapy(name:string, dru:drug, dose:float).
class therapeuticStrategy(patient:patient, therapy:therapy,startDate:date, nDay:integer).

```

The previous classes are a fragment of a medical ontology inherent (breast) neoplasm cares and are used to model the clinical process shown in Section 14.1.1. Class `primarySited_neoplasm` shows the ability to specify user-defined classes as attribute types (i.e. `site:bodyRegion`). Class `drug` has a list-type attribute `admRoute:[string]` representing possible Route of administration for a drug (for example inhalation, nasal, oral, parenteral). Relation schemas expressing medical knowledge can be declared by using the following syntax:

```

relation suffers (patient:patient, disease:disease).
relation relatedDrug (dis:disease, dru:drug).
relation sideEffect (dru:drug, effect:string).
relation classifiedAs (dis:disease, c:code).

```

Relation `suffer` asserts diseases suffered by a patient. Relations `relatedDrug` and `sideEffect` associates respectively drugs to a diseases and side effects to drugs. Moreover, relation `classifiedAs` enables users to query the ontologies by using codes defined in the original medical ontologies.

Ontology and Process Instances

Clinical process instances are expressed by ontology instances and created exclusively during process execution. Classes instances (objects) are defined by their *oid* (that starts with #) and a list of attributes. Instances obtained by executing the running example, are shown in the following.

```

#1:neoplasm_process(name:"Breast Neoplasm").
#2:therapy_administration_process(name:"Therapy Administration").
#1_1:acceptance_node(name:"Acceptance", container:#1, start_time:6580, end_time:16580,
    tasks:[#1_1_1], handler:#27).
#1_2:anamnesis_node(name:"Anamnesis", container:#1, start_time:16570, end_time:26580,
    nodes:[#1_2_1, #1_2_2, #1_2_3])
#1_2_3:recentPathological_anamnesis_node(name:"Recent Pathological Anamnesis", container:#1,
    start_time:19580, end_time:26570,tasks:[#1_2_3_1],handler:#27).
...

```

As described in section 14.1.1, instance of `anamnesis_node #1_2` is composed by a set of anamnesis activities represented by means of their *id*. The object `#1_2_3` belongs to `#1_2`. Objects `#1_1`, `#1_2_3` are tasks executed in custom and manual decision node and are stored as their attributes. When execution arrives in a task node or in a manual decision node, task instances is created and the user input is stored as values of the task attributes. Some tasks related to task nodes are shown in the following.

```

#1_1_1:acceptance_form(name:"Acceptance", patient:#21, acc_date:#data_089).
#1_2_3_1:pathology_form(name:"Recent Pathology", disease:#neoB_01).

```

For example, `acceptance_form` and `object` is obtained by a form filled by an oncologist. It contains an instance of patient class.

Transition and decision tuples, created during the process execution, are shown in the following. In the example, the `decision` is obtained as a manual choice of an oncologist, but instances of decisions could be automatically generated by means of reasoning tasks.

```
transition(name:"Acceptance-Anamnesis",from:#1_0, to:#1_1).
decision(name:"More Clinical Tests requested - No",from:#1_4, to:#1_6).
```

By considering the running example, instances for the classes `bodyRegion`, `breast_primarySited_neoplasm`, for the subclasses of `drug` and `code`, can be obtained by importing them from already existing medical ontologies and can be declared as follows:

```
#A01.236: bodyRegion(name:"breast").
#neoB_01: breast_primarySited_neoplasm(descr:"Malignant neoplasm of breast", site:#A01.236,
zone:"Nipple and areola").
#L02BG03: enzymeInhibitors(name:"Anastrozole", ddd:1, unit:mg, admRoute:["oral"], notes:"").
#L02AA04: estrogens(name:"Fosfestrol", ddd:0.25, unit:g, admRoute:["oral","parenteral"],
notes:"").
#icd10_C50.0: icd10Code(c:"C50.0", chapter:2, block:"C", category:"50", subCat:"0").
#mesh08_C04.588.180: mesh08Code(c:"C04.588.180",category:"C", subCat:"04").
```

The object having *id* `#neoB_01`, is an instance of the `breast_primarySited_neoplasm` class. Its attributes `descr` and `zone` (which type is `string`) have respectively value "Malignant neoplasm of breast" and "Nipple and areola", whereas the attribute `site` has value `#A01.236` that is an *id* representing an instance of the class `bodyRegion`. Tuples expressing medical knowledge can be declared by using the following syntax:

```
suffer (pat:#21, dis:@neoB_01).
relatedDrug (dis:@C50.9, dru:@L02BG03).
sideEffect (dru:@L02BG03, effect:"Chest pain").
sideEffect (dru:@L02BG03, effect:"Shortness of breath").
classifiedAs (dis:@neoB_01, c:@icd10_C50.0).
classifiedAs (dis:@neoB_01, c:@mesh08_C04.588.180).
```

The tuple of the relation `suffer` asserts that the patient `@p_002` suffers of the disease `@neoB_01`. The same diseases is classified in the ICD10-CM with identifier code `@icd10_C50.0`, and is stored in Mesh tree structure with identifier code `@mesh08_C04.588.180`. By means of the relation `classifiedAs` an user is enabled to querying ontology concept referring to the correspondent identifiers.

Information Extraction

Information extraction tasks are applied to input Electronic Medical Records (EMRs) and risk reports coming from different hospital wards. An EMR is generally a flat text document (having usually 3 pages) written in Italian natural language. EMRs are weakly structured, for example, the personal data of the patient are in the top of the document, clinical events (e.g medical

exams, surgical operations, diagnosis, etc.) are introduced by a date. Risk reports, filled at the end of clinical process, are provided to patients by wards to acquire information about errors with or without serious outcomes, adverse events, near misses.

The goal of the application is to extract semantic metadata about oncology therapies and errors with temporal data. The application extracts personal information (name, age, address), diagnosis data (diagnosis time, kind of tumor, body part affected by the cancer, cancer progression level), care and therapies information. Extracted information are exploited to construct, for each cared patient, an instance of lung cancer clinical process. Acquired process instances are analyzed by means of data and process mining techniques in order to discover if errors happen following patterns in phases of drugs prescription, preparation or administration.

The application has been obtained by exploiting a Semantic Model inherent to lung cancer that contains: (i) concepts and relationships referred to the disease, its diagnosis, cares in term of surgical operations and chemotherapies with the associated side effects. Concepts related to persons (patients), body parts and risk causes are also represented. All the concepts related to the cancer come from the ICD9-CM diseases classification system, whereas the chemotherapy drugs taxonomy, is inspired at the Anatomic Therapeutic Chemical (ATC) classification system. (ii) a set of descriptors enabling the automatic acquisition of the above mentioned concepts from Electronic Medical Records (EMRs). In the following a piece of the medical Semantic Model that describes (and allows to extract) patient name, surname, age and disease is shown.

It is noteworthy that in this example keywords, e.g. CONTAIN and SEBPY are used. CONTAIN is syntactic sugar used to represent the containment, only E direction is used, so they are not indicated SEBPY represents a separator between each pair of recognized object.

```
class anatomy ().
  class bodyRegion (bp:string) isa {anatomy}.
  class organ isa {body_part}.
    lung: organ("Lung").
    <lung>-><X:token(), matches(X,"[Ll]ung")>.
    ...
  ...
class disease (name:string).
  tumor: disease("Tumor").
  <tumor>-><X:token(), matches(X,"[Tt]umor")>.
  cancer: disease("Cancer").
  <cancer>-><X:token(), matches(X,"[Cc]ancer")>.
  ...
relation synonym (d1:disease,d2:disease)
  synonym(cancer,tumor).
  ...
class body_part_desease () isa {disease}.
  lung_cancer: body_part_desease("Lung cancer").
  <lung_cancer>-><diagnosis_section> CONTAIN <lung> & <X:desease(),synonym(cancer,X)>
  ...
collection class patient_data (){}
  collection class patient_name (name:string){}
    <patient_name(Y)> -> <T:token(),defBy(T, "name:")> <X:token()> {Y := X;}
```



```

SEPBY <X:space()>.
collection class patient_surname (surname:string){}
<patient_surname(Y)> ->
<X:hiStr(),matches(X,"sur(?:name)?:"> <X:token()> {Y:=X;} SEPBY <X:space()>.
collection class patient_age (age:integer){}
<patient_age(Y)>-><X:token(),matches(X,"age:"> <Z:token()>{Y := $str2int(Z);}
SEPBY <X:space()>.
...
collection class patient_data (name:string, surname:string,
                               age:integer, diagnosis:body_part_disease){}
<patient_data(X,Y,Z,lung_cancer)> ->
<hospitalization_section> CONTAIN <P:patient_name(X1)>{X:=X1}
& <P:patient_surname(Y1)>{Y:=Y1} & <P:patient_age(Z1)>{Z:=Z1} & <lung_cancer>.
...

```

The classes `diagnosis_section` and `hospitalization_section` used in the above descriptors represent text paragraphs containing personal data and diagnosis data recognized by proper descriptors that aren't shown for lack of space. The extraction mechanism can be considered in a WOXM fashion: Write Once eXtract Many, in fact the same descriptors can be used to enable the extraction of metadata related to patient affected by `lung_cancer` in unstructured EMRs that have different arrangement. Moreover, descriptors are obtained by automatic writing methods (as happens, for example, for the cancer and tumor concepts) or by visual composition (as happens for `patient_data`)

Metadata extracted by using the Semantic Model are stored as collection class instances into a knowledge base. For the simple piece of Semantic Model shown above the extraction process generates the following `patient_data` class instance for an EMR: "@1": `patient_data("Mario","Rossi","70",lung_cancer)`.

The application is able to process many EMRs and risk reports in a single execution and to store extracted metadata in XML format.

Reasoning Over Schemas and Instances

Since the OCKRF is built on top of DLP+ [109, 158], integrity constraints and complex inference rules can be expressed over schemas and instances respectively by means of *axioms* and *reasoning tasks*. For example, the following axiom prevents the prescription of a drug to a patient that has an allergy to a particular constituent of the drug.

```

::-therapyStrategy(patient:P, therapy:T, drug:D),hasActivePrinciple(drug:D,con Constituent:C),
allergy(patient:P,actPrin:C).

```

Axioms could be, also, used for: (i) specify constraints about transitions behavior. For example, the axiom "`::-P:process(), not start_node(container:P).`" expresses that a `start_node` must exist for each process. Constraints express, also, that a transition links nodes belonging to the same process, and corresponds to an effective edge of the process model as shown in the following:

```

::-transition(from:N1,to:N2), N1:node(container:P1), N2:node(container:P2), P1!=P2.
::-transition(from:N1,to:N2), N1:node(start_time:ST1), N2:node(start_time:ST2), ST1>=ST2.

```

```

::P:neoplasm_process(), transition(from:N1,to:N2), N1:acceptance_node(container:P),
not N2:anamnesis_node(container:P).
...

```

A reasoning task can be used to throw a medical error when the prescribed dose exceed the recommended dose based on individual characteristics (i.e. age and weight) of the interested patient. Such a check is useful when a `therapeutic_strategy_form` is created while `therapeutic_strategy_definition_node` is active.

```

ID:drugPrescription_medicalError(relativeTo:TASK,timestamp:TIME,msg:MSG):-
TASK:chemotherapeutic_strategy_form(strategy:STR),STR:therapeuticStrategy(patient:P,
therapy:T),
P:patient(bornDate:DATE,weight:W), @age(date,AGE), T:therapy(dru:DRUG,dose:DOSE),
recommendedDose(drug:DRUG, dose:RD, minAge:MA, MinWeight:MW), AGE<MA, W<MW, DOSE>RD,
MSG:="Prescribed dose " + DOSE + "exceed recommend dose " + RD, @newID(ID), @now(TIME).

```

The generated prescription error event must be properly handled in the process, for example an error message is visualized by means of a GUI to the physician.

```

ID:view_msg(method:"exception.jar", msg:MSG):-
X:drugPrescription_medicalError(relativeTo:TASK, timestamp:TIME, msg:MSG), @newID(ID).

```

Queries can be also used for exploring clinical processes ontologies in a semantic fashion. For instance `malNeoplasm_f_patient(patient:P)?` returns every female patients suffering of any malignant neoplasm (e.g `P=#21`, `P=#34` ids are given for answer), where `malNeoplasm_f_patient(patient:P)`:

```

malNeoplasm_f_patient(patient:P):- P:patient(sex:#F),suffer(patient:P,disease:D),
D:malignant_neoplasm().

```

During process execution, process and ontology instances are extracted from clinical records and acquired, and stored in a knowledge base. The execution can be monitored by running (over clinical process schemas and instances) reasoning tasks that implements error and risk rules. Extracted information could be analyzed for identifying main causes of medical errors, high costs and, potentially, suggesting clinical processes improvement able to enhance cost control and patient safety.

14.2 Records Management

In this section is summarized the application of visual wrapping capabilities describe in this thesis to the extraction and acquisition of metadata from administrative records like invoices. Companies of all industries, in fact, are overwhelmed by huge amounts of electronic and paper administrative records like: invoices, order and purchase forms, waybills (consignment notes), etc. having different layouts and presentation templates. To manually manage administrative documents limits visibility and efficiency in business processes and exposes companies to financial and operational risks. For instance, lack

of visibility on invoices can create discrepancies, lost discount opportunities, overpayments, among other risks.

Visual information extraction capabilities made available by approaches, techniques, languages and algorithms described in this thesis allow for automatically acquiring information and metadata contained in administrative documents in order to correctly include them in business processes execution. This way errors and risks due to loss of information can be limited and better performances in terms of efficiency and efficacy can be obtained. Users can choose in a flexible way information and metadata to acquire from administrative documents. For instance, dates, total amounts, VAT codes, items, supplier codes, prizes per unit, quantities, etc. can be automatically identified in invoices and extracted. Visual Wrappers for administrative documents can be automatically learned. Then the wrapper can be easily refined (if needed) and validated in few clicks. Learned wrappers can be reused for extracting automatically information from thousand pages.

The screenshot shows a PDF document titled 'marchigiani 1.pdf' in Adobe Acrobat Pro Extended. A red-bordered 'Visual Wrapper' is applied to the invoice content. The wrapper highlights the header information and a table of items. The header includes the company name 'RICEVUTA BANCARIA 60 GG. F.M.', the invoice number 'FTI / 1288', and the date '31/07/2007'. The table lists items with columns for date, description, quantity, unit price, and total amount.

DATE	DESCRIPTION	QUANTITY	UNIT PRICE	TOTAL AMOUNT	VAT
02/07/2007	PRODOTTI FINITI	1,000	1.450,00	1.450,00	20
03/07/2007	PRODOTTI FINITI	1,000	1.090,00	1.090,00	20
04/07/2007	PRODOTTI FINITI	1,000	350,00	350,00	20
06/07/2007	PRODOTTI FINITI	1,000	350,00	350,00	20

Fig. 14.2. A Visual Wrapper for an Invoice

For example, invoices can be automatically handled. As shown in Figure 14.2 for each different invoice template, a visual wrapper can be learned and edited. By visual wrappers, a set of user defined information (e.g. addresses, items, dates, total amounts and so on) can be automatically extracted from invoices and stored in structured form as depicted in Figure 14.3.

14.3 Document Understanding

In this section is briefly explained an application aimed at extracting information and metadata from the official bulletin of region government. Spatial and

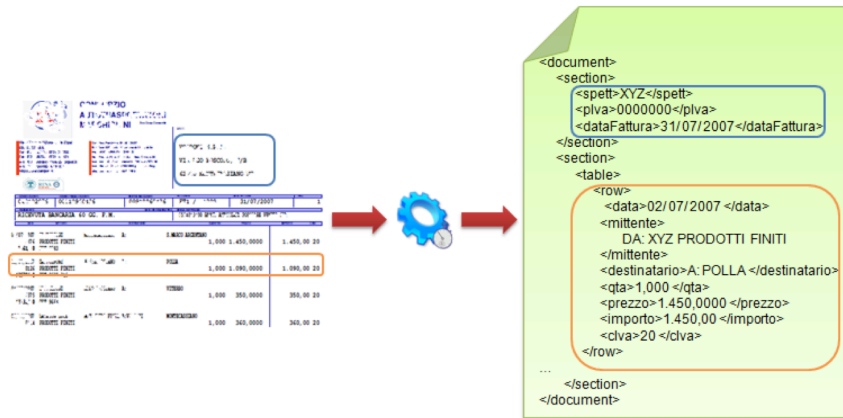


Fig. 14.3. The Result of the Visual Wrapping Process for an Invoice

semantic information extraction techniques defined in this thesis, in fact, allow Document Layout Analysis and Recognition techniques that enable document decomposition and splitting. In particular, by applying capabilities of the XONTO systems Titles, Sections, Paragraphs, Columns, Tables, Sentences, Image, Graphical Areas etc. can be automatically recognized and extracted. For example, a law contained in the official bulletin of a regional government can be identified on the base of their contents and split in its part as shown in Figure 14.4. Then each part can be analyzed and annotated over concepts like: *public funding* for a *cultivation* in the *agricultural industry*. Furthermore, the law can be transformed in another format like HTML or DOC in order to publish it on a Web site or Edit it in a word processor.

14.4 News Paper Review

In this section is shortly described the application of visual Web wrapping methods, presented in this thesis, for enabling automatic information extraction from Web source that makes available specific information to users. For instance, knowledge bases containing company profiles or product catalogs, where each product is described by its features, can be automatically created by using wrappers automatically learned from the Web sources. In Figure 14.5 and 14.6 it is shown how wrappers for news of on-line newspapers can be learned, so news can be automatically extracted, stored in XML form and reused in mash-up applications.

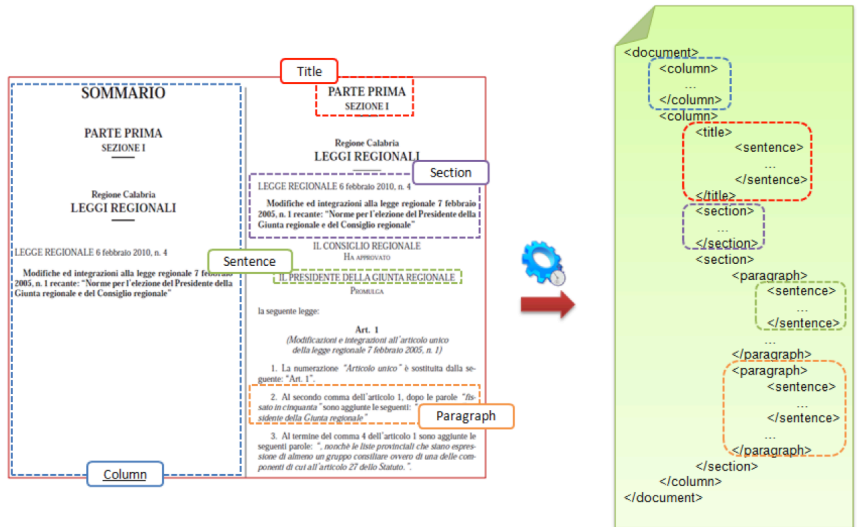


Fig. 14.4. The Result of the Visual Wrapping Process for an Official Bulletin

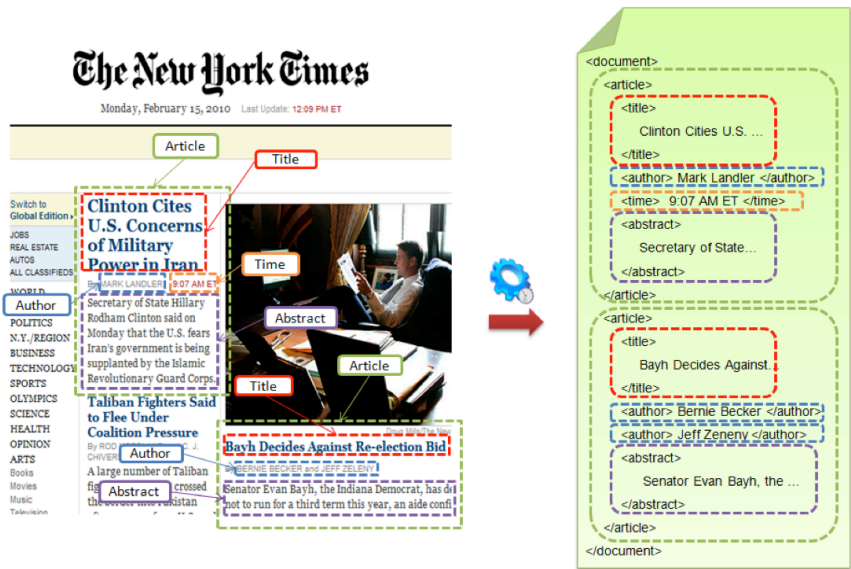


Fig. 14.5. The Result of the Visual Wrapping Process for the NYT Web Site

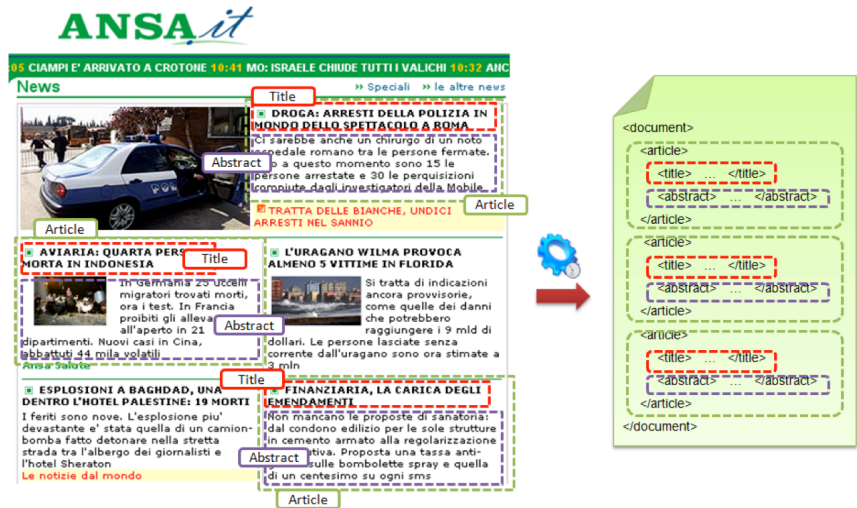


Fig. 14.6. The Result of the Visual Wrapping Process for the ANSA Web Site

Part IV

Conclusion and Future Work

*I am enough of an artist to draw freely upon my imagination.
Imagination is more important than knowledge.
Knowledge is limited.
Imagination encircles the world.
~ Albert Einstein ~*

Conclusion and Future Work

15.1 Conclusion and Future Work

This thesis has addressed the problem of extracting information from presentation-oriented documents by using spatial features of layouted content items and, also, semantic features of presented information made available by ad hoc domain knowledge representations. The rest of this chapter will summarize the content of this work, remark the main contributions and briefly discuss future trends in the considered research field.

15.1.1 Content Summary

Motivation that draw this thesis work have been laid out in Chapter 1 of Part I by investigating some possible applications of information extraction approaches and technologies, and major open research challenges in the field. A detailed description of existing information extraction and query languages for presentation-oriented documents has been given in Part II. Related work has been classified on the base of the document format on which existing techniques can be applied and by considering the level of automation of approaches in literature. Another classification direction has been the possibility to use prior domain knowledge for defining wrappers and extraction rules.

In Part III has been given the detailed description of original contribution in the filed of information extraction obtained during the PhD course. In particular, in Chapter 9 has been presented the spatial documents models that allows for representing in unified way PODs having both HTML and PDF internal representation. In Chapter 10 has been described the PDF-TREX approach based on document understanding techniques that led to automatic table recognition and extraction, and the SILA approach for wrapper induction for Web documents. In Chapter 11 has been presented the SXPath language, that extends XPath and allows for querying PODs by means of spatial navigation constructs, and the spatial grammars formalism that allows for

expressing wrapper as a set of productions of a CFG extended in order to consider spatial and semantic relations among elements displayed on PODs. In Chapter 12 has been presented ontology-based information extraction capabilities made available by the XONTO approach. This approach exploits a particular version of spatial grammars that provide the ability to model and handle spatial relations, and the ability to consider domain knowledge about the information to extract modeled in a knowledge base. In Chapters 8, 13, and 14 have been described the system that can be built by using approaches described in previous chapters, the visual capabilities made available by SILA, SXPath and PDF-TREX approaches, and some applications to real world scenarios respectively.

15.1.2 Contributions

The research done during the PhD and described in this thesis has led to the following main contributions:

- Definition of a unique spatial model for representing PODs having different internal formats. By such a model content items in documents having HTML and PDF internal encodings can be represented along with their spatial features and relations.
- Definition of techniques, based on heuristics and machine learning algorithms, for automatic recognition and extraction of tables from PODs. Automatic table extraction algorithms are very well performing as show by experiments. From the comparison with existing approaches results that defined algorithm constitute the current state of the art in this field.
- Definition of wrappers learning approaches based only on spatial PODs features. Defined techniques, in fact exploit only the spatial relations among content items in order to learn wrappers for repetitive records contained in Deep Web Pages. Such techniques do not require preprocessing of the HTML tags aimed at inferring spatial arrangement, at the contrary they use directly the layouted Web page. Experiments show that the SILA algorithm performs better than approaches available in literature.
- Definition of formal languages aimed at describing and querying PODs. In particular, the SXPath language allows user to write very intuitive pattern similar to XPath expressions and capable to exploit spatial features of PODs resulting in a more simple query mechanism for this kind of documents as experiments shown. The spatial grammars formalism allow user to write wrapper a set of spatial productions that can be intuitively designed by considering only what the user see on the screen. Both SXPath and spatial grammars open the doors to novel wrapper learning techniques that considers the spatial structure of PODs and the semantic of contents.
- Definition of methods, languages and approaches for knowledge representation that support information extraction and semantic annotation by enabling the description and automatic recognition of concepts contained in PODs.

Future Trends

Information extraction is a research field with a lot of potential [52, 39], it can be expected to grow in different directions. A first future work direction comes from current new user needs. Nowadays Web users, in fact, need objects (having a clear semantics) retrieved from different Web pages and arranged in a comprehensible way. Hence, the Web should be viewed as a set of linked objects (like persons, events, products, companies, locations, etc.) described by attributes and relations among them. This new vision has been originally introduced by Berners-Lee and takes the name of Web of Objects (WOO) or Web of Concepts (WOC) [12, 44, 148]. A second future work direction is related to semantic Web search and mash-up. In this direction users need intelligent applications capable to retrieve contents on the base of their semantics and combine retrieved contents on the base of their semantic relations. For example, by means of these features search engines will be able to provide to users that search for a conference, flights, hotels, car rental services, colleagues that will participate to the same conference by mashing-up semantically related objects coming from different sources [162].

References

1. Acid Tests, <http://www.acidtests.org>. *Web Standards Project*.
2. S. Adali, M. L. Sapino, and V. S. Subrahmanian. An algebra for creating and querying multimedia presentations. *Multimedia Syst.*, 8(3):212–230, 2000.
3. B. Adelberg. Nodose - a tool for semi-automatically extracting structured and semistructured data from text documents. In *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, SIGMOD '98, pages 283–294, New York, NY, USA, 1998. ACM.
4. Adobe Systems Incorporated, www.adobe.com/devnet/pdf/pdf_reference.html. *PDF Reference and Specification*.
5. M. Aiello, C. Monz, L. Todoran, and M. Worring. Document understanding for a broad class of documents. *IJDAR*, 2002.
6. J. Aitken. Learning Information Extraction Rules: An Inductive Logic Programming approach. In *Proceedings of the 15th European Conference on Artificial Intelligence*, pages 355–359, 2002. <http://citeseer.ist.psu.edu/586553.html>.
7. H. Alblas. Introduction to attributed grammars. In *Proceedings on Attribute Grammars, Applications and Systems*, pages 1–15, London, UK, 1991. Springer-Verlag.
8. J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
9. O. Altamura, F. Esposito, and D. Malerba. Transforming paper documents into xml format with wisdom++. *International Journal of Document Analysis and Recognition*, 4:2001, 2000.
10. A. Arasu and H. Garcia-Molina. Extracting structured data from web pages. In *SIGMOD '03: Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 337–348, New York, NY, USA, 2003. ACM.
11. Y. Aumann, R. Feldman, Y. Liberzon, B. Rosenfeld, and J. Schler. Visual information extraction. *Knowl. Inf. Syst.*, 10(1):1–15, 2006.
12. R. Baeza-Yates. Searching the web of objects. In *ICOOB*, 2010.
13. P. Balbiani, J.-F. Condotta, and L. F. d. Cerro. A model for reasoning about bidimensional temporal relations. In *Proc. of KR-2008*, pages 124–130, 1998.
14. P. Balbiani, J.-F. Condotta, and L. F. d. Cerro. A new tractable subclass of the rectangle algebra. In *IJCAI*, pages 442–447, 1999.
15. M. Banko, M. J. Cafarella, S. Soderl, M. Broadhead, and O. Etzioni. Open information extraction from the web. In *In IJCAI*, pages 2670–2676, 2007.

16. R. Baumgartner, S. Flesca, and G. Gottlob. The elog web extraction language. In *Proceedings of the Artificial Intelligence on Logic for Programming, LPAR '01*, pages 548–560, London, UK, 2001. Springer-Verlag.
17. R. Baumgartner, S. Flesca, and G. Gottlob. Visual web information extraction with lixto. In *VLDB*, pages 119–128, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
18. R. Baumgartner, W. Gatterbauer, and G. Gottlob. Web data extraction. *Encyclopedia of Database Systems*. Springer-Verlag, 2009.
19. R. Baumgartner, G. Gottlob, and M. Herzog. Scalable web data extraction for online market intelligence. *VLDB*, 2(2):1512–1523, 2009.
20. M. Benedikt, W. Fan, and G. Kuper. Structural properties of xpath fragments. *Theor. Comput. Sci.*, 336(1):3–31, 2005.
21. M. Benedikt and C. Koch. Xpath leashed. *ACM Comput. Surv.*, 41(1):1–54, 2008.
22. T. Berners-Lee, W. Hall, J. A. Hendler, K. O'Hara, N. Shadbolt, and D. J. Weitzner. A framework for web science. *Found. Trends Web Sci.*, 1(1):1–130, 2006.
23. M. A. Bhatti and A. Ahmad. Pdf to html conversion: Having a usable web document. In *Digital Information Management*, 2006.
24. S. Boag, A. Berglund, D. Chamberlin, J. Siméon, M. Kay, J. Robie, and M. F. Fernández. XML path language (XPath) 2.0. W3C recommendation, W3C, January 2007. <http://www.w3.org/TR/2007/REC-xpath20-20070123/>.
25. S. Boag, D. Chamberlin, M. Fernández, D. Florescu, J. Robie, and J. Siméon. XQuery 1.0: An XML query language. W3C Working Draft, 2005.
26. A. Bozzon, M. Brambilla, S. Ceri, and P. Fraternali. Liquid query: multi-domain exploratory search on the web. In *Proceedings of the 19th international conference on World wide web, WWW '10*, pages 161–170, New York, NY, USA, 2010. ACM.
27. T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. Extensible markup language (xml) 1.0 (fourth edition). World Wide Web Consortium, Recommendation REC-xml-20060816, August 2006.
28. D. Buttler, L. Liu, and C. Pu. A fully automated object extraction system for the world wide web. In *ICDCS*, 2001.
29. A.-Y. S. B. C. and S. J. Te xquery: A full-text search extension to xquery. In *WWW*, 2004.
30. D. Cabeza and M. Hermenegildo. Distributed www programming using (ciao-)prolog and the pillow library. *Theory Pract. Log. Program.*, 1(3):251–282, 2001.
31. A. Cali and D. Martinenghi. Querying the deep web. In *EDBT '10: Proceedings of the 13th International Conference on Extending Database Technology*, pages 724–727, New York, NY, USA, 2010. ACM.
32. M. Califf and R. Mooney. Relational learning of pattern-match rules for information extraction. In *AAAI*, 1999.
33. J. Carme, M. Ceresna, and M. Goebel. Web wrapper specification using compound filter learning. In *IADIS International Conference WWW/Internet*, 2006.
34. X. Carreras, I. Chao, L. PadrÛ, and M. PadrÛ. Freeling: An open-source suite of language analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC'04)*, 2004.

35. B. t. Cate and C. Lutz. The complexity of query containment in expressive fragments of xpath 2.0. *J. ACM*, 56(6):1–48, 2009.
36. D. Chamberlin, J. Robie, and D. Florescu. Quilt: an XML Query Language for Heterogeneous Data Sources. In *Lecture Notes in Computer Science*. Springer-Verlag, December 2000.
37. C.-H. Chang, M. Kaye, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *TKDE*, 18(10):1411–1428, 2006.
38. C.-H. Chang and S.-C. Lui. Iepad: information extraction based on pattern discovery. In *WWW '01: Proceedings of the 10th international conference on World Wide Web*, pages 681–688, New York, NY, USA, 2001. ACM.
39. L. Chiticariu, Y. Li, S. Raghavan, and F. R. Reiss. Enterprise information extraction: recent developments and open challenges. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 1257–1258, New York, NY, USA, 2010. ACM.
40. P. Cimiano and J. Völker. Text2onto. In *NLDB*, pages 227–238, 2005.
41. E. Cortez, A. S. da Silva, M. A. Gonçalves, and E. S. de Moura. Ondux: on-demand unsupervised learning for information extraction. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 807–818, New York, NY, USA, 2010. ACM.
42. V. Crescenzi, G. Mecca, and P. Merialdo. Roadrunner: Towards automatic data extraction from large web sites. In *VLDB '01: Proceedings of the 27th International Conference on Very Large Data Bases*, pages 109–118, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc.
43. H. Cunningham, K. Bontcheva, V. Tablan, and D. Maynard. *General Architecture for Text Engineering*. <http://www.gate.ac.uk>, 2003.
44. N. Dalvi, R. Kumar, B. Pang, R. Ramakrishnan, A. Tomkins, P. Bohannon, S. Keerthi, and S. Merugu. A web of concepts. In *PODS '09: Proceedings of the twenty-eighth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–12, New York, NY, USA, 2009. ACM.
45. E. Dantsin, T. Eiter, G. Gottlob, and A. Voronkov. Complexity and expressive power of logic programming. In *IEEE Conference on Computational Complexity*, pages 82–101, 1997.
46. A. Das Sarma, A. Jain, and D. Srivastava. I4e: interactive investigation of iterative information extraction. In *SIGMOD '10: Proceedings of the 2010 international conference on Management of data*, pages 795–806, New York, NY, USA, 2010. ACM.
47. T. Declerck, C. Federmann, B. Kiefer, and H.-U. Krieger. Ontology-based information extraction and reasoning for business intelligence applications. In A. Dengel, K. Berns, T. M. Breuel, F. Bomarius, and T. Roth-Berghofer, editors, *KI*, volume 5243 of *Lecture Notes in Computer Science*, pages 389–390. Springer, 2008.
48. G. Della Penna, D. Magazzeni, and S. Orefice. Visual extraction of information from web pages. *J. Vis. Lang. Comput.*, 21(1):23–32, 2010.
49. A. Deutsch, M. Fernandez, D. Florescu, A. Levy, and D. Suciu. A Query Language for XML. In *Proceedings of the 8th Int. World Wide Web Conf. (WWW'8), Toronto, May 1999*, pages 1155–1169. Elsevier Science B.V., 1999.
50. W. Dilger. Automatic translation with attribute grammars. In *Proceedings of the 8th conference on Computational linguistics*, pages 397–404, Morristown, NJ, USA, 1980. Association for Computational Linguistics.

51. H. Djean and J.-L. Meunier. A system for converting pdf documents into structured xml format. In *Workshop on Document Analysis Systems*, 2006.
52. A. Doan, R. Ramakrishnan, and S. Vaithyanathan. Managing information extraction: state of the art and research directions. In *SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 799–800, New York, NY, USA, 2006. ACM.
53. R. W. W. Draft. *XQuery 1.0 and XPath 2.0 Full-Text 1.0*. <http://www.w3.org/TR/xpath-full-text-10-requirements/>.
54. W. W. Draft. *XQuery 1.0 and XPath 2.0 Full-Text 1.0*. <http://www.w3.org/TR/xpath-full-text-10/>.
55. P. Eades and K. Sugiyama. How to draw a directed graph. *J. Inf. Process.*, 13(4):424–437, 1990.
56. J. Earley. An efficient context-free parsing algorithm. *Commun. ACM*, 13(2):94–102, 1970.
57. J. Eisner and J. Blatz. Program transformations for optimization of parsing algorithms and other weighted logic programs. In S. Wintner, editor, *Proceedings of FG 2006: The 11th Conference on Formal Grammar*, pages 45–85. CSLI Publications, 2007.
58. T. Eiter, G. Gottlob, and H. Mannila. Disjunctive Datalog. *ACM Transactions on Database Systems*, 22(3):364–418, 1997.
59. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, D. W. Lonsdale, Y.-K. Ng, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. *Data Knowl. Eng.*, 31(3):227–251, 1999.
60. D. W. Embley, D. M. Campbell, Y. S. Jiang, S. W. Liddle, Y.-K. Ng, D. Quass, and R. D. Smith. Conceptual-model-based data extraction from multiple-record web pages. In *Data Knowl. Eng.*, 1999.
61. G. Erbach. Bottom-up earley deduction. *CoRR*, cmp-lg/9502004, 1995.
62. B. Fazzinga, S. Flesca, A. Tagarelli, S. Garruzzo, and E. Masciari. A wrapper generation system for pdf documents. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, pages 442–446, New York, NY, USA, 2008. ACM.
63. C. D. Fellbaum. *WordNet – An Electronic Lexical Database*. MIT Press, 1998.
64. G. Fiumara. Automated information extraction from web sources: a survey. 2007.
65. S. Flesca, S. Garruzzo, E. Masciari, and A. Tagarelli. Wrapping pdf documents exploiting uncertain knowledge. In *CAiSE*, 2006.
66. S. Flesca, E. Masciari, and A. Tagarelli. A fuzzy logic approach to wrapping pdf documents. *TDKE*, 2010.
67. R. Florian, H. Jing, N. Kambhatla, and I. Zitouni. Factorizing complex models: a case study in mention detection. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 473–480, Morristown, NJ, USA, 2006. Association for Computational Linguistics.
68. G. Fourny, D. Kossmann, T. Kraska, M. Pilman, and D. Florescu. Xquery in the browser. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, SIGMOD '08, pages 1337–1340, New York, NY, USA, 2008. ACM.
69. G. Fourny, M. Pilman, D. Florescu, D. Kossmann, T. Kraska, and D. McBeath. Xquery in the browser. In *Proceedings of the 18th international conference on*

- World wide web*, WWW '09, pages 1011–1020, New York, NY, USA, 2009. ACM.
70. D. Freitag. Machine learning for information extraction in informal domains. In *Machine Learning*, 2000.
 71. D. Freitag and N. Kushmerick. Boosted wrapper induction. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 577–583. AAAI Press, 2000.
 72. W. Gatterbauer, P. Bohunsky, M. Herzog, B. Krüpl, and B. Pollak. Towards domain-independent information extraction from web tables. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 71–80, New York, NY, USA, 2007. ACM.
 73. D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distrib. Parallel Databases*, 3(2):119–153, 1995.
 74. G. Gottlob and C. Koch. Monadic datalog and the expressive power of languages for web information extraction. *J. ACM*, 51:74–113, January 2004.
 75. G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The litxo data extraction project: back and forth between theory and practice. In *PODS*, pages 1–12, 2004.
 76. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing xpath queries. In *VLDB*, pages 95–106, 2002.
 77. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing xpath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.
 78. G. Gottlob, C. Koch, R. Pichler, and L. Segoufin. The complexity of xpath query evaluation and xml typing. *J. ACM*, 52(2):284–335, 2005.
 79. M. Grigni, D. Papadias, and C. Papadimitriou. Topological inference. In *IJCAI*, pages 901–906, 1995.
 80. R. Grishman and B. Sundheim. Message understanding conference - 6: A brief history. In *Proceedings of the International Conference on Computational Linguistics*, 1996.
 81. J. Ha, R. M. Haralick, and I. T. Phillips. Recursive x-y cut using bounding boxes of connected components. In *ICDAR*, 1995.
 82. J. Hammer, J. McHugh, and H. Garcia-Molina. Semistructured data: The tsimmi experience. In *ADBIS*, pages 1–8, 1997.
 83. A. Harth, M. Janik, and S. Staab. *Handbook of Semantic Web*. Springer, 2011.
 84. T. Hassan. Graphwrap: a system for interactive wrapping of pdf documents using graph matching techniques. In *DocEng '09: Proceedings of the 9th ACM symposium on Document engineering*, pages 247–248, New York, NY, USA, 2009. ACM.
 85. T. Hassan. User-guided wrapping of pdf documents using graph matching techniques. In *ICDAR '09: Proceedings of the 2009 10th International Conference on Document Analysis and Recognition*, pages 631–635, Washington, DC, USA, 2009. IEEE Computer Society.
 86. T. Hassan. Towards a common evaluation strategy for table structure recognition algorithms. In *DocEng*, 2010.
 87. T. Hassan and R. Baumgartner. Intelligent text extraction from pdf documents. In *CIMCA-IAWTIC, Volume 02*, pages 2–6, Washington, DC, USA, 2005. IEEE Computer Society.

88. T. Hassan and R. Baumgartner. Table recognition and understanding from pdf files. In *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, pages 1143–1147, Washington, DC, USA, 2007. IEEE Computer Society.
89. C. D. L. Higuera and J. Oncina. Inferring deterministic linear languages. In *COLT '02: Proceedings of the 15th Annual Conference on Computational Learning Theory*, pages 185–200, London, UK, 2002. Springer-Verlag.
90. K. Hofmann and W. Weerkamp. *Web corpus cleaning using content and structure*. C. Fairoon, H. Naerts, A. Kilgarriif, and G. de Schryver, 2007.
91. J. L. Hong, E.-G. Siew, and S. Egerton. Information extraction for search engines using fast heuristic techniques. *Data Knowl. Eng.*, 69(2):169–196, 2010.
92. C.-N. Hsu and M.-T. Dung. Generating finite-state transducers for semi-structured data extraction from the web. *Inf. Syst.*, 23:521–538, December 1998.
93. G. Huck, P. Fankhauser, K. Aberer, and E. J. Neuhold. Jedi: Extracting and synthesizing information from the web. In *COOPIS '98: Proceedings of the 3rd IFCIS International Conference on Cooperative Information Systems*, pages 32–43, Washington, DC, USA, 1998. IEEE Computer Society.
94. U. Irmak and T. Suel. Interactive wrapper generation with minimal user effort. In *Proceedings of the 15th international conference on World Wide Web, WWW '06*, pages 553–563, New York, NY, USA, 2006. ACM.
95. ISO8879:1986. Information processing – Text and Office Systems – Standard Generalized Markup Language (SGML). Standard No. ISO 8879:1986, International Organization for Standardization, 1986.
96. K. Ito and Y. Tanaka. A visual environment for dynamic web application composition. In *HYPertext*, pages 184–193. ACM, 2003.
97. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
98. JBoss. jpdL. jBPM Process Definition Language (jPDL). <http://www.jboss.org/jbossjbpm/jpdL/>.
99. D. Johnson. *What is tagged PDF?* <http://www.planetpdf.com/enterprise/article.asp?ContentID=6067>, 2005.
100. D. Johnson. *Microsoft Office Word 2007 Rich Text Format (RTF) Specification.*, 2007.
101. T. Kieninger and A. Dengel. An approach towards benchmarking of table structure recognition results. In *ICDAR*, pages 1232–1236, 2005.
102. J. Kong, K. Zhang, and X. Zeng. Spatial graph grammars for graphical user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 13(2):268–307, 2006.
103. R. Krishnamurthy, Y. Li, S. Raghavan, F. Reiss, S. Vaithyanathan, and H. Zhu. Systemt: a system for declarative information extraction. *SIGMOD Rec.*, 37(4):7–13, 2008.
104. B. Kriüpl, M. Herzog, and W. Gatterbauer. Using visual cues for extraction of tabular data from arbitrary html documents. In *WWW*, pages 1000–1001, New York, NY, USA, 2005. ACM.
105. N. Kushmerick, D. S. Weld, and R. Doorenbos. Wrapper induction for information extraction. In *Proc. IJCAI-97*, 1997.
106. A. H. F. Laender, B. Ribeiro-Neto, and A. S. da Silva. Debye - date extraction by example. *Data Knowl. Eng.*, 40(2):121–154, 2002.
107. A. H. F. Laender, B. A. Ribeiro-Neto, A. S. da Silva, and J. S. Teixeira. A brief survey of web data extraction tools. *SIGMOD Rec.*, 31(2):84–93, 2002.

108. T. Lee, L. Sheng, T. Bozkaya, N. H. Balkir, Z. M. Özsoyoglu, and G. Özsoyoglu. Querying multimedia presentations based on content. *TKDE*, 11(3):361–385, 1999.
109. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlw system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.
110. K. Lerman, L. Getoor, S. Minton, and C. Knoblock. Using the structure of web sites for automatic segmentation of tables. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 119–130, New York, NY, USA, 2004. ACM.
111. Y. Li and K. Bontcheva. Hierarchical, perceptron-like learning for ontology-based information extraction. In *Proceedings of the 16th international conference on World Wide Web, WWW '07*, pages 777–786, New York, NY, USA, 2007. ACM.
112. L. Libkin. *Elements Of Finite Model Theory*. SpringerVerlag, 2004.
113. R. Lima, B. Espinasse, and F. Freitas. An adaptive information extraction system based on wrapper induction with pos tagging. In *SAC '10: Proceedings of the 2010 ACM Symposium on Applied Computing*, pages 1815–1820, New York, NY, USA, 2010. ACM.
114. B. Liu, R. Grossman, and Y. Zhai. Mining data records in web pages. In *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 601–606, New York, NY, USA, 2003. ACM.
115. L. Liu, C. Pu, and W. Han. Xwrap: An xml-enabled wrapper construction system for web information sources. In *Proceedings of the 16th International Conference on Data Engineering*, pages 611–, Washington, DC, USA, 2000. IEEE Computer Society.
116. W. Liu, X. Meng, and W. Meng. Vide: A vision-based approach for deep web data extraction. *IEEE Trans. on Knowl. and Data Eng.*, 22(3):447–460, 2010.
117. Y. Liu, K. Bai, P. Mitra, and C. L. Giles. Improving the table boundary detection in pdfs by fixing the sequence error of the sparse lines. In *ICDAR, ICDAR '09*, pages 1006–1010, Washington, DC, USA, 2009. IEEE Computer Society.
118. Y. Liu, P. Mitra, and C. L. Giles. Identifying table boundaries in digital documents via sparse line detection. In *Proceeding of the 17th ACM conference on Information and knowledge management, CIKM '08*, pages 1311–1320, New York, NY, USA, 2008. ACM.
119. S. J. Løvborg. Declarative programming and natural language, 2007.
120. B. Ludäscher, R. Himmeröder, G. Lausen, W. May, and C. Schleppehorst. Managing semistructured data with florid: a deductive object-oriented perspective. *Inf. Syst.*, 23(9):589–613, 1998.
121. G. F. Luger. *Artificial Intelligence: Structures and Strategies for Complex Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1997. Preface By-Stubblefield,, William A.
122. J. Madhavan, S. R. Jeffery, S. Cohen, X. . Dong, D. Ko, C. Yu, A. Halevy, and G. Inc. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, 2007.
123. A. Maedche, G. Neumann, and S. Staab. Bootstrapping an ontologybased information extraction system, 2002.

124. M. Marx and M. de Rijke. Semantic characterizations of navigational xpath. *SIGMOD Rec.*, 34(2):41–46, 2005.
125. L. McDowell and M. J. Cafarella. Ontology-driven information extraction with ontosyphon. In *International Semantic Web Conference (ISWC)*, pages 428–444, 2006.
126. S. Mir, S. Staab, and I. Rojas. Unsupervised approach for acquiring ontologies and rdf data from online life science databases. In *ESWC*, 2010.
127. A. A. Muchnik. One application of real-valued interpretation of formal power series. *Theor. Comput. Sci.*, 290(3):1931–1946, 2003.
128. I. Muslea. Extraction patterns for information extraction tasks: A survey. In *In AAAI-99 Workshop on Machine Learning for Information Extraction*, pages 1–6, 1999.
129. I. Muslea, S. Minton, and C. A. Knoblock. Hierarchical wrapper induction for semistructured information sources. *Autonomous Agents and Multi-Agent Systems*, 4(1-2):93–114, 2001.
130. G. Nagy and S. Seth. Hierarchical representation of optically scanned documents. In *ICPR*, 1984.
131. G. Nagy, S. Seth, and M. Viswanathan. A prototype document image analysis system for technical journals. *Computer*, 25(7):10–22, 1992.
132. I. Navarrete and G. Sciavicco. Spatial reasoning with rectangular cardinal direction relations. In *ECAI*, pages 1–9, 2006.
133. K. Nicholas. *Wrapper induction for information extraction*. PhD thesis, 1997. Chairperson-Weld, Daniel S.
134. OASIS. *OASIS, Open Document Format for Office Applications (OpenDocument)*. <http://docs.oasis-open.org/office/v1.1/OS/>, 2007, 2007.
135. E. Oro, F. Riccetti, and M. Ruffolo. Viquel: A spatial query language for presentation-oriented documents. In *ICTAI*, 2010.
136. E. Oro, F. Riccetti, and M. Ruffolo. A spatial approach for extracting information from presentation-oriented documents. In *ICAART*, 2011.
137. E. Oro and M. Ruffolo. Description ontologies. In *ICDIM*, pages 369–374, 2008.
138. E. Oro and M. Ruffolo. Towards a system for ontology-based information extraction from pdf documents. In *OTM Conferences (2)*, pages 1482–1499, 2008.
139. E. Oro and M. Ruffolo. Xonto: An ontology-based system for semantic information extraction from pdf documents. In *ICTAI (1)*, pages 118–125, 2008.
140. E. Oro and M. Ruffolo. Pdf-trex: An approach for recognizing and extracting tables from pdf documents. In *ICDAR*, pages 906–910, 2009.
141. E. Oro and M. Ruffolo. Towards a semantic system for managing clinical processes. In *ICEIS (2)*, pages 180–187, 2009.
142. E. Oro, M. Ruffolo, and D. Saccà. Combining attribute grammars and ontologies for extracting information from pdf documents. In *SEBD*, pages 153–160, 2009.
143. E. Oro, M. Ruffolo, and D. Saccà. Ontology-based information extraction from pdf documents with xonto. *International Journal on Artificial Intelligence Tools (IJAIT)*, 18(5):673–695, 2009.
144. E. Oro, M. Ruffolo, and D. Saccà. A semantic clinical knowledge representation framework for effective health care risk management. In *BIS*, pages 25–36, 2009.
145. E. Oro, M. Ruffolo, and S. Staab. Sxpath - extending xpath towards spatial querying on web documents. *PVLDB*, 4(2):129–140, 2010.

146. E. Oro, M. Ruffolo, and F. Valentini. Sila: A spatial instance learning approach from deep web pages. In *ICAR-CNR Technical Report*, 2010.
147. T. J. Ostrand, M. C. Paull, and E. J. Weyuker. Parsing regular grammars with finite lookahead. *Acta Inf.*, 16:125–138, 1981.
148. Oxford University, Domain-centric intelligent automated data extraction methodology, <http://www.w3.org/html/>. *DIADEM- Domain-centric Intelligent Automated Data Extraction Methodology*.
149. L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. A (sub)graph isomorphism algorithm for matching large graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(10):1367–1372, 2004.
150. N. K. Papadakis, D. Skoutas, K. Raftopoulos, and T. A. Varvarigou. Stavies: A system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques. *TKDE*, 17(12):1638–1652, 2005.
151. P. Parys. Xpath evaluation in linear time with polynomial combined complexity. In *PODS*, pages 55–64. ACM, 2009.
152. F. Pereira, A. Rajaraman, S. Sarawagi, W. Tunstall-Pedoe, G. Weikum, and A. Halevy. Answering web questions using structured data: dream or reality? *Proc. VLDB Endow.*, 2:1646–1646, August 2009.
153. A. Pivk, P. Cimiano, Y. Sure, M. Gams, V. Rajkovic, and R. Studer. Transforming arbitrary tables into logical form with tartar. *Data Knowl. Eng.*, 60(3):567–595, 2007.
154. B. Popov, A. Kiryakov, A. Kirilov, D. Manov, D. Ognyanoff, and M. Goranov. Kim - semantic annotation platform. In *International Semantic Web Conference*, pages 834–849, 2003.
155. L. D. Raedt, A. Kimmig, and H. Toivonen. Problog: a probabilistic prolog and its application in link discovery. In *IJCAI*, pages 2468–2473. AAAI Press, 2007.
156. J. Raposo, A. Pan, M. Álvarez, J. Hidalgo, and A. Vi na. The wargo system: Semi-automatic wrapper generation in presence of complex data access modes. In *DEXA '02: Proceedings of the 13th International Workshop on Database and Expert Systems Applications*, pages 313–320, Washington, DC, USA, 2002. IEEE Computer Society.
157. J. Renz. *Qualitative spatial reasoning with topological information*. Springer, 2002.
158. F. Ricca and N. Leone. Disjunctive logic programming with types and objects: The dlv+ system. *J. Applied Logic*, 5(3):545–573, 2007.
159. E. Riloff. Automatically generating extraction patterns from untagged text. In *Proceedings of the thirteenth national conference on Artificial intelligence - Volume 2*, AAAI'96, pages 1044–1049. AAAI Press, 1996.
160. G. Rozenberg and A. Salomaa, editors. *Handbook of formal languages, vol. 1: word, language, grammar*. Springer-Verlag New York, Inc., New York, NY, USA, 1997.
161. D. Rus and K. Summers. Geometric algorithms and experiments for automated document structuring. *Mathematical and Computer Modelling*, 26:55–83, 1997.
162. M. B. S. Ceri. *Search Computing Challenges and Directions*. Springer, 2010.
163. H. Saggion, A. Funk, D. Maynard, and K. Bontcheva. Ontology-based information extraction for business intelligence. In *ISWC/ASWC*, pages 843–856, 2007.

164. A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *DKE*, 36(3):283–316, 2001.
165. J. Schürmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberländer. Document analysis-from pixels to contents. *Proceedings of the IEEE*, pages 403–421, 2002.
166. D. Shasha, J. T. L. Wang, and R. Giugno. Algorithmics and applications of tree and graph searching. In *PODS*, pages 39–52, 2002.
167. W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd international conference on Very large data bases, VLDB '07*, pages 1033–1044. VLDB Endowment, 2007.
168. W. Shen, A. Doan, J. F. Naughton, and R. Ramakrishnan. Declarative information extraction using datalog with embedded extraction predicates. In *Proceedings of the 33rd international conference on Very large data bases, VLDB '07*, pages 1033–1044. VLDB Endowment, 2007.
169. K. Simon and G. Lausen. Viper: augmenting automatic information extraction with visual perceptions. In *Proceedings of the 14th ACM international conference on Information and knowledge management, CIKM '05*, pages 381–388, New York, NY, USA, 2005. ACM.
170. D. Simovici and R. Tenney. *Theory of Formal Languages with Applications*. World Scientific, Singapore, 1999.
171. S. Soderland, C. Cardie, and R. Mooney. Learning information extraction rules for semi-structured and free text. In *Machine Learning*, pages 233–272, 1999.
172. S. Staab and R. Studer. *Handbook on Ontologies*. Springer Publishing Company, Incorporated, 2nd edition, 2009.
173. K. Sudo, S. Sekine, and R. Grishman. An improved extraction pattern representation model for automatic ie pattern acquisition. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1, ACL '03*, pages 224–231, Morristown, NJ, USA, 2003. Association for Computational Linguistics.
174. B. ten Cate and M. Marx. Navigational xpath: calculus and algebra. *SIGMOD Rec.*, 36(2):19–26, 2007.
175. B. ten Cate and M. Marx. Axiomatizing the logical core of xpath 2.0. *Theor. Comp. Sys.*, 44(4):561–589, 2009.
176. B. ten Cate and L. Segoufin. Xpath, transitive closure logic, and nested tree walking automata. In *PODS*, pages 251–260. ACM, 2008.
177. J. Turmo, A. Ageno, and N. Catal. Adaptive information extraction. *ACM Computing Surveys*, 38(2), 2006.
178. S. Vadrevu, F. Gelgi, and H. Davulcu. Information extraction from web pages using presentation regularities and domain knowledge. *World Wide Web*, 10(2):157–179, 2007.
179. M. Vargas-vera, E. Motta, J. Domingue, S. B. Shum, and M. Lanzoni. Knowledge extraction by using an ontology-based annotation tool. In *In K-CAP 2001 workshop on Knowledge Markup and Semantic Annotation*, pages 5–12, 2001.
180. W3C, <http://www.w3.org/Style/CSS/>. *Cascading Style Sheets*.
181. W3C, <http://www.w3.org/DOM/>. *Document Object Model (DOM)*.
182. W3C, <http://www.w3.org/html/>. *HTML*.
183. W3C, <http://www.w3.org/XML/Query/>. *XML Query (XQuery)*, 1.0 edition.
184. W3C, <http://www.w3.org/TR/xpath>. *XML Path Language (XPath) Version 1.0*, 1.0 edition, November 1999.

185. P. Wadler. Two semantics for xpath. Draft: <http://homepages.inf.ed.ac.uk/~wadler/papers/xpath-semantics>, 2000.
186. Y. Wang. *Document analysis: a table structure understanding and zone content classification*. PhD thesis, 2002.
187. G. Weikum and M. Theobald. From information to knowledge: harvesting entities and relationships from web sources. In *PODS '10: Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems of data*, pages 65–76, New York, NY, USA, 2010. ACM.
188. K. Wich. Exponential ambiguity of context-free grammars. In G. Rozenberg and W. Thomas, editors, *Proceedings of the 4th International Conference on Developments in Language Theory*, pages 125–138. World Scientific, Singapore, July 2000.
189. D. C. Wimalasuriya and D. Dou. Ontology-based information extraction: An introduction and a survey of current approaches. *Journal of Information Science*, 2009.
190. M. M. Wood, S. J. Lydon, V. Tablan, D. Maynard, and H. Cunningham. Populating a database from parallel texts using ontology-based information extraction. In *NLDB*, pages 254–264, 2004.
191. W. A. Woods. Transition network grammars for natural language analysis. *Commun. ACM*, 13(10):591–606, 1970.
192. F. Wu and D. S. Weld. Autonomously semantifying wikipedia. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, pages 41–50, New York, NY, USA, 2007. ACM.
193. R. Yangarber, R. Grishman, P. Tapanainen, and S. Huttunen. Automatic acquisition of domain knowledge for information extraction. In *Proceedings of the 18th conference on Computational linguistics - Volume 2*, pages 940–946, Morristown, NJ, USA, 2000. Association for Computational Linguistics.
194. B. Yildiz, K. Kaiser, and S. Miksch. Pdf2table: A method to extract table information from pdf files. In *IICAI*, pages 1773–1785, 2005.
195. F. Yuan, B. Liu, and G. Yu. A study on information extraction from pdf files. In *ICMLC*, 2005.
196. R. Zanibbi, D. Blostein, and R. Cordy. A survey of table recognition: Models, observations, transformations, and inferences. *Int. J. Doc. Anal. Recognit.*, 7(1):1–16, 2004.
197. Y. Zhai and B. Liu. Web data extraction based on partial tree alignment. In *WWW '05: Proceedings of the 14th international conference on World Wide Web*, pages 76–85, New York, NY, USA, 2005. ACM.
198. Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *TKDE*, 18(12):1614–1628, 2006.
199. Y. Zhai and B. Liu. Extracting web data using instance-based learning. In *WWW*, pages 113–132, 2007.
200. S. Zhang, J. Li, H. Gao, and Z. Zou. A novel approach for efficient supergraph query processing on graph databases. In *EDBT '09: Proceedings of the 12th International Conference on Extending Database Technology*, pages 204–215, New York, NY, USA, 2009. ACM.
201. H. Zhao, W. Meng, Z. Wu, V. Raghavan, and C. Yu. Fully automatic wrapper generation for search engines. In *Proceedings of the 14th international conference on World Wide Web*, WWW '05, pages 66–75, New York, NY, USA, 2005. ACM.