WebMeaning

Bringing the Knowledge on the Web to Software Agents

Author: José Luis Arjona
Advisor: Dr. Rafael Corchuelo

May, 13 2003
José Luis Arjona, D.N.I number 44.218.635-P.

AUTHORIZATION

I hereby declare that I am the sole author of the pre–PHD entitled:

WebMeaningBringing the Knowledge on the Web to Software Agents.

José Luis Arjona
Lecturer in Computer Engineering
Department of Computer Languages and Systems
University of Seville

May, 13 2003
WebMeaningBringing the Knowledge on the Web to Software Agents.

by
José Luis Arjona

This is to certify that I have examined the above pre-PhD and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the pre-PhD examination committee can be made.

Rafael Corchuelo
Reader in Computer Engineering

May, 13 2003
To Sina,
The Woman of my Dreams
Acknowledgments

First of all, I would like to thank my advisor Rafael Corchuelo for his guidance and encouragement through my research and for believing in me.

I would like to thank Miguel for making this research possible.

I would like to thank Deib, Antonio, Kino, Oztabio, Jperez and Benabides for the time we spent together and for their interesting feedback on my research.

I would like to thank José Antonio, Victor and Vicente for letting me participate in the matters of languages. They give me the freedom and trust to enjoy teaching.

I would like to thank Paco, Raúl, Amador, Jesús A., Isabel, Bea and the rest of colleagues of the department LSI for creating and maintaining so good teaching environment.

Finally, and most important, I would also like to thank my parents for bringing me into this world and for supporting me unconditionally in the pursuit of my life.

J.L.A.
May, 13 2003
Contents

1 Introduction ........................................ 1
   1.1 The World Wide Web: An Enormous Repository of Information ....... 1
   1.2 Why Accessing to the Knowledge on the Web is a Problem ........... 2
   1.3 Contributions ........................................ 4
   1.4 Pre–Phd Overview ..................................... 6

2 Related work ........................................ 7
   2.1 Preliminaries ........................................ 7
       2.1.1 Information Retrieval and Information Extraction .......... 8
       2.1.2 A Classification of Web Pages ........................... 8
       2.1.3 Single–Slot and Multi–Slot Extraction ....................... 9
       2.1.4 Information, Knowledge and Semantic Information .......... 9
   2.2 Extracting Information from the Web ........................ 11
       2.2.1 Artificial Intelligence: Inductive Wrappers ............... 12
       2.2.2 Database Community: Information Mediators ............... 20
   2.3 Extracting Semantic Information from Web Pages ................. 22
       2.3.1 Squeal ............................................. 22
       2.3.2 WebOntEx ........................................ 22
   2.4 Extracting Knowledge from the Web .......................... 24
       2.4.1 The Semantic Web .................................... 24
       2.4.2 Omnibase and START .................................. 26
       2.4.3 The WEB→KB Project .................................. 28
   2.5 Summary .............................................. 29
       2.5.1 Extracting Information from the Web ....................... 29
### Contents

2.5.2 Extracting Semantic Information from the Web .......... 31  
2.5.3 Extracting Knowledge from the Web .................... 31  

3 A Description of WebMeaning .................. 33  
3.1 A Method to Construct Knowledge Channels ............... 33  
3.1.1 Generating the syntactic wrapper ..................... 35  
3.1.2 Generating the Syntactic Verifier ...................... 35  
3.1.3 Defining the Semantic Description .................. 36  
3.1.4 Specifying the Navigation to Web Pages .............. 37  
3.2 The WebMeaning Reference Architecture .................. 38  
3.3 Summary ............................................. 41  

4 A Model for WebMeaning .................... 43  
4.1 Syntactic Wrappers .................................. 43  
4.2 Syntactic Verification .................................. 45  
4.3 Semantic Wrappers .................................. 46  
4.3.1 Dealing with Knowledge ............................ 46  
4.3.2 A formal Specification for Semantic Wrappers .......... 48  
4.3.3 Semantic Verification .................................. 51  
4.4 Navigating through the Web ............................. 52  
4.4.1 A Case Study ...................................... 52  
4.4.2 Navigational State Machine ......................... 52  
4.4.3 Pagers and Indexers .................................. 54  
4.5 Bringing all together: Knowledge Channels .............. 59  
4.6 Summary ............................................. 63  

5 Conclusions and Future Work ................ 65  
5.1 Conclusions ........................................... 65  
5.2 Future Work ........................................... 66  
5.3 Practical Validation of WebMeaning .................... 67  
5.3.1 Helping a Spanish Political Party to Attract Voters ...... 67  
5.3.2 Web Site Personalisation ............................ 68  
5.3.3 Knowledge Base about Web Services .................. 69
Contents

A Well–formed formula 71

B Curriculum 75

B.1 Education .............................................. 75
B.2 Professional Experience ................................ 75
B.3 Publications .............................................. 76
    B.3.1 Journals ........................................ 76
    B.3.2 Book Chapters ................................... 77
    B.3.3 Conferences .................................... 79
B.4 Official Research Projects .............................. 83
B.5 I+D Organisation Activities ............................. 85
B.6 I+D with enterprises .................................. 86
List of Figures

1.1 A web page with information about scores in a golf championship 3
2.1 Information about some golfers .................................. 10
2.2 Related work overview ........................................ 11
2.3 Inductive wrapper operation ................................... 12
2.4 Example of extraction rules in RAPIER ....................... 14
2.5 Example of extraction rules in SRV ........................... 15
2.6 Example of extraction rules in WHISK ....................... 17
2.7 Example of extraction rules in WIEN ........................ 18
2.8 Example of extraction rules in SoftMealy ..................... 19
2.9 Example of extraction rules in STALKER .................... 21
2.10 Example of queries using Squeal ............................. 23
2.11 The XML DTD for a company schema. ....................... 25
2.12 Semantic web evolution .................................... 26
2.13 Example of queries in the START system .................... 27
2.14 Example of the WEB→KB system ......................... 30
3.1 Overview of the method to construct knowledge channels. 34
3.2 A semantic wrapper. ..................................... 36
3.3 Proposed architecture ........................... 39
3.4 Broker Interaction Protocol in AUML ......................... 40
4.1 An example of a web page containing an iterator and a pager ... 55
4.2 An Example of Doom tree for a web page from Amazon ...... 60
List of Tables

2.1 Summary of inductive wrappers systems ......................... 31
Table of Programmes

3.1 A piece of code representing a navigational state machine ........ 38
Chapter 1

Introduction

Internet is so big, so powerful and pointless that for some people it is a complete substitute for life.

Andrew Brown

This chapter aims at describing and motivating the problem in which we are working. It is organised as follows: Next Section presents the world wide web as an information repository; then, Section 1.2 shows the main problems that appear when we are trying to extract the knowledge from today’s web; Section 1.3 describes the contributions we have made to the field of knowledge extraction; finally, Section 1.4 describes how this pre–dissertation is organized.

1.1 The World Wide Web: An Enormous Repository of Information

In recent years, the design of reference architectures for systems distributed on the Internet has attracted an increasing number of researchers and practitioners who have focused on platforms, languages, middlewares or interoperability concerns. The main reason for such a great interest is that this network
has experienced a rapid shift from information and entertainment to electronic commerce, which has gained importance and grown exponentially [Kalmbach and Palmer, 2002, Reynolds and Mofazali, 2000, Tsichritzis, 1998].

A major challenge for marketplace participants has become sifting through an unwieldly amount of information to find useful products and services. Fortunately, the technology has evolved and the Internet has matured to a point in which sophisticated new generation agents exist. They enable efficient, comprehensive searches on the vast web information repository, and can circumvent some problems related to slow Internet access and free up prohibitively expensive surf time by operating in the background. In order for such agents to be reliable and interoperable, they must be able to understand the information the web contains, which is difficult because it is mostly available in human-readable forms that lack formalised semantics [Berners-Lee et al., 2001], and it is likely to change its structure unexpectedly [Brewington and Cybenko, 2000, Francisco-Revilla et al., 2001, Lim et al., 2001, Starr et al., 1996].

This pre-phd presents a framework called WebMeaning whose main goal is to enhance the way agents interact on the web. Thus, WebMeaning allows agents to have easy access to the information they need from the web and its semantics. It associates semantics with the information extracted, which improves agent interoperability; it can also surf the web and automatically deal with changes to the structure of a web page, which improves adaptability; furthermore, it achieves a complete separation between the data extraction procedure and the logic or base functionality an agent encapsulates.

1.2 Why Accessing to the Knowledge on the Web is a Problem

We illustrate the problems of extracting knowledge from the web by means of a simple real example in which we are interested in extracting information about the score of golfers in a PGA championship. This information was given at http://www.golfweb.com. Figure 1.1 shows a web page from this site. If we were interested in extracting the information automatically, the following issues would arise immediately:

- The implied meaning of the terms that appear in this page can be easily interpreted by humans, but there is not a reference to the ontology that
1.2. Why Accessing to the Knowledge on the Web is a Problem

Figure 1.1: A web page with information about scores in a golf championship.
describes them precisely, which complicates communication and inter-operability amongst agents [Berners-Lee et al., 2001, 1992].

- The layout and the aspect of a web page may change unexpectedly. For instance, it is usual that web sites incorporate Christmas banners in December, which does not change the meaning of the information they provide, but may invalidate unexpectedly the automatic extraction methods used so far [Brewington and Cybenko, 2000, Francisco-Revilla et al., 2001, Lim et al., 2001, Starr et al., 1996].

- The access to the page that contains the information in which we are interested may involve navigating through a series of intermediate pages, e.g., login or index pages. Furthermore, this information may spread over a set of pages.

1.3 Contributions

In this pre–Phd, we describe the WebMeaning framework. It provides mechanisms to associate semantics with the information extracted, to deal with changes to the structure of a web page, to delegate the knowledge extraction procedure to specialised agents called knowledge channels, and to surf the web. This way, using WebMeaning enables the design and implementation of clean, reusable, understandable agents.

The following significant contributions we have made to address the problems identified in previous section:

- A method to construct knowledge channels, or KCs for short. They are agents [Wooldridge and Jennings, 1995] that allow to separate the extraction of knowledge from the logic of an agent, and they are able to react to knowledge inquiries (reactivity) from other agents (social ability), and act in the background (autonomy) to maintain a local knowledge base (KB) with knowledge extracted from a web site (proactivity). In order to allow for semantic interoperability, the knowledge they manage references a number of concepts in a given application domain that are described by means of ontologies.

KCJs extract knowledge from the web using semantic wrappers, which are a natural extension to current techniques for extracting information from
1.3. Contributions

today’s non-semantic web to deal with knowledge on the web. A semantic wrapper gives semantics to the information extracted by these techniques; this task is carried out by an algorithm called semantic translator that uses a semantic description that relates the information to be extracted with the ontology that brings it meaning. Thus, the semantic translator allows us to translate the information extracted by the syntactic wrapper to knowledge.

The problem of changes in a web page that may invalidate unexpectedly the automatic extraction methods used so far, is overcome by using syntactic verification algorithms, they check the extracted information to insure that it is correctly extracted.

WebMeaning also provides a framework to define the navigation to the web pages that contain the information of interest. In this framework the navigation is defined as a state machine, where a state references a set of web pages; and a transition from State 0 to State 1 references the process of navigating from pages that State 0 defines to pages that State 1 defines. Thus, the problem of web navigation is solved in WebMeaning by executing the machine that represents the navigation.

- A reference architecture [Arjona et al., 2002b] to ease how software agents have access to the knowledge extracted. It incorporates a broker agent that has information about registered knowledge channels (yellow pages). This way, the agents that need to use the extracted knowledge need not be aware of the existence of different KCs, which can thus be adapted, created or removed from the system transparently.

Additionally, we report on three case studies in which the WebMeaning framework was valuable.

1. To personalise web sites based on users’ profiles [Arjona et al., 2002a]. The information from the web that is interesting to a user can be extracted automatically and displayed in a suitable form.

2. To develop a web agent that helps a Spanish political party decide if it should organise an outdoors meeting to attract voters. Data mining techniques establish the agent decision rules and our framework feed them with information from the web. [Corchuelo et al., 2002]

3. To extract information about web services descriptions and to maintain a knowledge base about them. An application can query this knowledge.
1.4 Pre-Phd Overview

This pre-Phd is organised as follows:

Next chapter glances at other proposals. It first introduces some concepts related to information and knowledge extraction; some proposals from information extraction, semantic information extraction and knowledge extraction are presented then and compared with WebMeaning.

Chapter 3 presents the WebMeaning framework. It begins with the description of a method to construct intelligent applications to extract the knowledge residing on web pages; then, a reference architecture is presented to access the knowledge extracted.

Chapter 4 presents knowledge channels formally. It begin presenting specifications of concepts related with syntactic information extraction, namely: syntactic wrapper and syntactic verification; these terms, are then extended to deal with web knowledge; it shows then the process of surfing the web as a execution of a navigation state machine; finally, all these ideas are brought together to define knowledge channels formally.

Chapter 5 provides an analysis of the WebMeaning framework. It summarises our main conclusions and discusses future research directions. It concludes with a description of three case studies in which the WebMeaning framework is valuable.
Chapter 2

Related work

Insanity: doing the same thing over and over again and expecting different results.

Albert Einstein, (1879 - 1955)

Related work can be characterised along three dimensions: extraction of information, extraction of semantic information, and extraction of knowledge from web. This chapter is organised to classify proposals in these dimensions. The first Section presents some concepts related with information and knowledge extraction; Section 2.2 presents different proposals to extract information from the web; Then, Section 2.3 shows proposals to extract semantic information from web pages; Section 2.4 describes some proposals to extract knowledge from the web; finally, Section 2.5 summarises the ideas in this chapter and compares them with WebMeaning.

2.1 Preliminaries

Before we go into further details, it is necessary to present some concepts related to information and knowledge extraction that are used through the rest of the Chapter.
2.1.1 Information Retrieval and Information Extraction

First, it is important to clarify the difference between information extraction and information retrieval. These terms are usually used erroneously to refer to the same idea, but they refer to two different tasks.

Information retrieval systems select a relevant subset of web pages from a large collection based on a user query; the user must then browse the returned web pages to get the desired information. Rather than to select a subset of web pages, the goal of information extraction is to extract relevant information from web pages.

Note that information extraction and information retrieval are complementary techniques, and usually they are used in combination since the information is extracted from web documents gathered using a retrieval system.

2.1.2 A Classification of Web Pages

Hsu and Dung (1998) [Hsu and Dung, 1998] presented a categorisation of web pages based on the structuredness of attributes that users want to extract. This categorisation is interesting because the techniques used to extract information depends on the class to which web pages belong. The classification is as follows:

- **Structured**: A web page is structured if it provides itemised information; that is, the information has a predefined, strict format. This means that each attribute in a tuple can be correctly extracted based on syntactic properties, such as delimiters and the order of attributes. Therefore, the techniques used in to extract information from structured web pages are quite simple if the format is known; otherwise, the format must be learned.

- **Semi-Structured**: A web page is classified as semi-structured if it may contain tuples with missing attributes, or attributes with multiple values, or attribute permutations, or exceptions. Usually, inductive techniques are used to extract information from semi-structured web pages; they use sample data to train an algorithm that generalises rules to extract information.

- **Unstructured**: A web page is considered unstructured if it is required...
linguistic knowledge to extract attributes correctly. In this case, the information to be extracted is in sentential form.

The techniques used to extract information from unstructured web pages are related with natural language. They typically obtain patterns that build on syntactic relations between words or semantic classes of words, in order to extract information automatically.

The techniques used in information extraction determine the kind of web pages that can be handled. For instance, if the system uses natural language processing techniques, it will work fine on unstructured, but not with web pages in which information is not in sentential form [Dale et al., 2000]. Furthermore, it is not easy to delimit the scope of a piece of data because of the HTML tags used to specify how to render it, which implies these techniques are not appropriate in general [Harmelen and Fensel, 1999] to extract from structured and semi–structured web pages.

2.1.3 Single–Slot and Multi–Slot Extraction

The information about golf scores that can be extracted from the web page in Figure 1.1 may include fields such as golfers’ names, scores and positions in a ranking. In information extraction terms, these fields are called slots. Thus, the information extraction task consist of filling slots with piece of data from the web page.

Information extraction systems perform single–slot or multi–slot information extraction. A system is able to perform single–slot extraction, when it can only extract isolated pieces of text from the web page. A system is able to perform multi–slot extraction if it can locate a pattern and searches recursively for similar patterns in the web page. For instance, in order to extract information about several golfers from the web page in Figure 1.1 we need to perform multi–slot extraction.

2.1.4 Information, Knowledge and Semantic Information

If we feed an information extraction system with the web page in Figure 1.1, it returns a frame as shown in Figure 2.1. This output is information; that is, a set of related facts (data).
Chapter 2. Related work

---

Golfer
Name : Rich Beem
Score : 278
Position: 1

Golfer
Name : Chris Riley
Score : 283
Position: 3

Golfer
Name : Tiger Woods
Score : 279
Position: 2

Golfer
Name : Justin Leonard
Score : 284
Position: 4

---

Figure 2.1: Information about some golfers.

Notice that the implied meaning of the terms that appear in this template can be easily interpreted by humans, but there is not a reference to the semantic information that describes them precisely, which makes it difficult for agents can understand its contents.

Ontologies are used to specify the meaning of information. Their aim is to specify concepts and relationships amongst them in a concrete domain. Thus, ontologies [Corcho and Gómez-Pérez., 2000] allows us to specify semantic information about the information to be extracted. For instance, next formula from a first-order logical language, can be used to represent the semantics of the information in figure 2.1. It defines concepts and relationships of interest amongst them in a golf championship domain.

\[
\forall x \cdot \text{Golfer}(x) \Rightarrow \text{Person}(x), \\
\forall x \cdot \exists y \cdot \text{Golfer}(x) \Rightarrow \text{Score}(x, y), \\
\forall x \cdot \exists y \cdot \text{Golfer}(x) \Rightarrow \text{Position}(x, y)
\]

This ontology may be used to annotate the information in Figure 2.1 and establishes a relationship between the information extracted and the ontology (knowledge). This way, an application using the golf championship ontology may understand the information in the frame; also communication and semantic interoperability amongst software applications is favoured [Berners-Lee et al., 2001, 1992]. For instance, the terms:

\[
\text{Golfer}(\text{Rich Beem}), \ \text{Score}(\text{Rich Beem, 278}), \ \text{Position}(\text{Rich Beem, 1})
\]
Figure 2.2: Related work overview.

means that Rich Beem is a golfer, and he ranks the first with 278 points.

There exists several proposals to extract information, semantic information and knowledge from the web. Figure 2.2 show this idea and present the systems we describe in this chapter.

2.2 Extracting Information from the Web

In this section we report on common approaches to extract information. They come from the field of artificial intelligence and of the databases community. This Section have been structured to show main systems developed in these communities.
Chapter 2. Related work

2.2.1 Artificial Intelligence: Inductive Wrappers

Wrappers [Eikvil, 1999] are one of the most popular mechanisms for extracting information from the web. Generally, a wrapper is an algorithm that translates the information represented in model $M_1$ into model $M_2$. In information extraction, they are able to translate the information in a web page ($M_1$) to a data structure ($M_2$) that can be used by software applications.

In the past, these algorithms were codified manually, using properties of web page’s, usually looking for strings that delimit the data that we want to extract. An important contribution to this field was provided by Kushmerick. He introduced induction techniques and defined a new class of wrappers called inductive wrappers. These inductive algorithms are components that use a number of extraction rules generated by means of automated learning techniques such as inductive logic programming, statistical methods, and inductive grammars. These rules set up a generic algorithm to extract information from similar pages automatically. Figure 2.3 shows this idea.

To generate the extraction rules, we have to feed the syntactic wrapper
2.2. Extracting Information from the Web

generator with a set of pairs of the following form:

\[ \{(p_1, t_1), (p_2, t_2), \ldots, (p_k, t_k)\}; \quad k \geq 1 \]

where \( p_i \) denotes a web page containing sample data, and \( t_i \) is the data that must be extracted from this web page. \( k \) depends on the structure of the web page and characteristic values can be found in the description that gives the wrapper’s author. With this information, we can apply the induction algorithm. The output is a set of extraction rules \( R_1, R_2, \ldots, R_m \) that we can use to set up the inductive wrapper. Boosted techniques [Freitag and Kushmerick, 2000] are proposed to improve the performance of the machine learning algorithm by repeatedly applying it to the training set with different example weightings.

Next, we describe the most relevant inductive wrappers systems (they all have over 50 references in citeseer\(^1\)), namely: RAPIER, SRV, WHISK, WIEN, SoftMealy and STALKER.

**RAPIER**

RAPIER [Califf and Mooney, 1998] stands for Robust Automated Production of Information Extraction Rules. It was developed by Mary Elaine Califf and Raymond J. Mooney at University of Texas at Austin. It is an efficient learning method that performs single-slot extraction on structured and semi-structured web pages. It can also deal with missing items and permutations (items that appear in different orders).

RAPIER takes pairs of documents and filled templates and induces extraction rules that directly extract fillers for the slots in the template. The inductive algorithm used by RAPIER is based on several Inductive Logic Programming (ILP) systems.

The extraction rules generated with RAPIER are based both on syntactic information (delimiters) and semantic information (content description). They are indexed by a template name and a slot name and have three parts: a **pre-filler** pattern that must match the text immediately following the information to be extracted (a right delimiter), a **filler** pattern that describes the structure of the information to be extracted, and a **post-filler** pattern that must match the text following the information to be extracted (a left delimiter).

\(^1\)Computer Science Papers NEC Research Institute (http://citeseer.nj.nec.com/cs).
Sample web page:

AI. C Programmer. 38-44K
Leading AI firm in need of
an energetic individual to fill the following position:

Filled template:

computer-science-job
  title : C Programmer
  salary: 38-44K
  area : AI

Extraction rule for area slot:

Pre--filler pattern:
  word: Leading
Filler Pattern:
  list: len: 2
tags: [nn,nns]
Post-Filler pattern:
  word: [firm, company]

Figure 2.4: Example of extraction rules in RAPIER.

Figure 2.4 shows a web page, the filled template used and the extraction pattern generated for the slot area. The sample web page is taken from an job posting domain. The extraction rule, generated by RAPIER means that to extract the area, we must look for the piece of text that is located between the pre–filler pattern (in this case the word Leading) and the post-filler pattern (the word firm or company). It is also required that the area slot has at most two words that were labeled “nn” or “nns” by the POS tagger [Brill, 1994] (one or two singular or plural common nouns)
Sample web pages:

Document 1: ... to purchase 4.5 mln Trilogy shares at ...
Document 2: ... acquire another 2.4 mln Roach shares at ...

Extraction rule:

Acquisition: length(<2),
    some(?A [capitalized true]),
    some(?A [next-token] all-lower-case true),
    some(?A [right-AN] wn-word 'stock').

Figure 2.5: Example of extraction rules in SRV.

SRV

SRV [Freitag, 1998] was developed by Dayne Freitag at the University of Carnegie Mellon and it stands for Sequence Rules with Validation. It is able to perform single-slot extraction on semi-structured and structured web pages. It can also deal with missing items and permutations.

The SRV system generates first-order logic extraction rules that are based on features that are simple (length, character type, ...) or relational (next-token, prev-token: next and previous token to the attribute to be extracted, ...). Using FOL to represent rules makes them very expressive. The learning consists of identifying and generalising the features found in the training examples. A drawback of SRV is that it needs a lot of data to generate the extraction rules.

Figure 2.5 shows two web pages given to SRV in order to extract the name of the companies that were target of an acquisition process. The extraction rule has the following meaning: the first and the second predicates indicate that a company name consists of a single capitalized word; the third predicate indicates that the next word is a lower-case word; last predicate uses features derived by the link grammar parser [Sleator and Temperley, 1993] and WordNet [Miller, 1995]. The “right—AN” refers to “right AN link” in a link grammar. It means that next word to the information to be extracted (A?) is one of the WordNet synonyms associated with the word stock.
WHISK

WHISK [Soderland, 1999] was developed by Stephen Soderland at the University of Washington. WHISK is a learning system that performs multi-slot extraction on structured, semi-structured and unstructured web pages. It can also deal with missing items and permutations; items permutation is achieved by feeding WHISK with sample data containing all the permutations of data to be extracted; this means that probably this process needs a large volume of training data.

The rules generated by WHISK are based on regular expression patterns that have two components: one that describes the context that makes a phrase relevant, and one that specifies the exact delimiters of the phrase to be extracted. Depending of the structure of the web pages, WHISK generates patterns that rely on either of the components: for unstructured web pages it uses context-based patterns; if a web page is structured then delimiter-based patterns are used; finally, for semi–structured web pages both of them are used.

Figure 2.6 shows a sample of WHISK. The sample web page is taken from an apartment rental domain. The extraction rule means that the wrapper must ignore all the characters in the text until it finds a digit (a predefined regular expression for digits from 0 through 9) followed by string BR string; it extracts that digit and fills the first extraction slot with it (Bedrooms). It then ignores the remaining characters until it reaches a dollar sign immediately followed by a number (a predefined regular expression for integer numbers). Then, it extracts the number filling the Price slot with it.

It is also possible to replace literals in a rule by semantic classes. A semantic class is a placeholder for any of the terms that defines it. For instance, we can replace BR with the semantic class Bedroom, which is defined as:

\[ \text{Bedroom} ::= (\text{br} | \text{brs} | \text{bdrm} | \text{bedrooms} | \text{bedroom}) \]

WIEN

WIEN [Kushmerick et al., 1997] was developed by Nicholas Kushmerick at the College Dublin University. It stands for Wrapper Induction Environment and was the first wrapper induction system. WIEN is a fast learning system that performs multi–slot extraction on structured web pages.

WIEN introduces several types of wrappers that assume that items are always in fixed, known order, and that web page have an HLRT organisation.
Sample web page:

Capitol Hill - 1 br twnhme.
D/W W/D. Pkg incl $675.
3 BR upper flr no gar. $995.
(206) 999-9999 <br>

A WHISK rule to extract numbers of bedrooms and prices:

ID :: 1
Pattern:: * (<Digit>) 'BR' * '$' (<Nmb>)
Output :: Rental {Bedrooms $1} {Price $2}

Output from the WHISK rule:

Rental:
    Bedrooms: 1
    Price  : 675
Rental:
    Bedrooms: 3
    Price  : 995

Figure 2.6: Example of extraction rules in WHISK.
Sample web pages:

Document 1: 1. Joe’s: (313)323-5545 2. LI’s: (406)545-2020

Extraction Rule:

```
*'.')(*)':*'( *)(*)'
```

Output:

Restaurant {Name @1}{AreaCode @2}

---

**Figure 2.7:** *Example of extraction rules in WIEN.*

This means that there is a Head delimiter, a set of Right and Left delimiter for each item to be extracted, and a Tail delimiter at the end. This makes that WIEN cannot handle permutations or missing items. The extraction rules generated by WIEN are similar to those generated by WHISK, the difference being that WIEN only uses delimiters that immediately precede and follow to the data to be extracted.

Figure 2.7 shows an example of WIEN. The sample web pages are taken from a web site that offers information about restaurants. The extraction rule is generated by WIEN to extract the name of a restaurant and its code. It means that wrapper ignores characters until the first occurrence of ‘.’ is found; it then extracts the name of a restaurant as the string between the ‘.’ found and the first ‘.’. Then, it ignores characters until a ‘)’ is found and extracts the string that ends at ‘)’ as the code number. Note that WIEN fails to extract information from D2 because of the format of phone number is different.

**SoftMealy**

SoftMealy [Hsu, 1998] was developed by Chun-Nan Hsu at the Arizona State University. It extracts data from web pages by learning wrappers specified as automata. It performs single-slot extraction on semi-structured and
2.2. Extracting Information from the Web

Sample web pages:

Document 1: 1. Joe’s: (313)323-5545 2. Li’s: (406)545-2020

Extraction rule:

```
*’.’(*) EITHER ’:’ (Nmb) ’-’
OR ’:’*’('(Nmb)’)
```

Output:

Restaurant {Name @1}{AreaCode @2}

**Figure 2.8:** Example of extraction rules in SoftMealy.

The wrappers generated using Softmealy are non-deterministic finite automata in which states represent the facts to be extracted and transitions amongst states represent contextual rules that define the separators between them. It allows both semantic classes and disjunctions, this way it works well with web pages having item permutations or missing items; however, in order to deal with item permutations, SoftMealy must be trained with samples that include each possible ordering of the items to be extracted.

Figure 2.8 shows the same example that we used to illustrate WIEN. The extraction rule means that a wrapper must ignore characters until the first occurrence of ’.’ is found, then it extract the name of a restaurant as the string between the ’.’ found and the first ’.’. If ’.’ is followed by a number, it extracts it as the area code, otherwise it ignores all characters until a ’(’ followed of a number and a ’)’ is found and it extract this number as the code number.

**STALKER**

STALKER [Muslea et al., 1998] is a wrapper induction algorithm for inducing extraction rules developed by Ion Muslea, Steve Minton and Craig
Knoblock at the University of Southern California. It performs hierarchical information extraction on structured and semi-structured web pages. It can deal with item permutations and missing items; to do that, it does not need to take all the permutations into account.

Hierarchical information extraction is necessary for web pages as the one presented in Figure 2.9. It gives information about restaurant-chain having restaurant located in several cities; for each city, the restaurant may have several locations, and for each location, it may have several phone numbers.

To address the problem of hierarchical information extraction, STALKER uses the Embedded Catalog Tree (ECT) formalism. ECT specifies the output schema for the extraction task, and it is used to guide the extraction process. For each node on ECT, STALKER generates extraction rules, plus an iteration rule if it is a LIST node.

The extraction process is performed in an hierarchical manner; for instance, to extract all the names of cities, it applies the extraction rule for City that means that it looks for the second \(<BR/>\) tag in the web page and it extracts everything until an \(<HR/>\) tag is found; then, to extract the names of cities it applies the iteration rule to the piece of text extracted by the extraction rule.

### 2.2.2 DataBase Community: Information Mediators

There are some related proposals, called information mediators, in the field of databases, e.g., TSIMMIS [García-Molina et al., 1995] and ARANEUS [G. Mecca and Atzeni., 1999]. Their goal is to integrate heterogeneous information sources such as traditional databases and web pages so that the user can work on them as if they were a homogeneous information source. The information from several sources are accessed via database-like queries.

The main with these proposals is that they lack a systematic way to extract information from the web because extraction rules need to be implemented manually, which makes them not scalable and unable to recover from unexpected changes on the web. Thus, we are not going to give further details about these proposals.
2.2. Extracting Information from the Web

Sample web page:

Name: Taco Bell <br><p><br>
- LA: 400 Pico; (213)323-5545,(800)222-1111. 211 Flower;(213)424-7645.<p>
- Venice: 20 Vernon;(310)888-1010.<p><hr>

Embedded Catalog Tree:

Document ::= Restaurant LIST(City)
City ::= CityName LIST(Location)
Location ::= Number Street LIST(Phone)
Phone ::= AreaCode PhoneNumber

Extraction rules:

Restaurant extraction rule: *‘Name:’(*)’<br>
LIST(City) extraction rule: *’<br>’*’<br>’(*)’<hr>
LIST(City) iteration rule: *’-’(*)’<p>
CityName extraction rule: *(*)’:’

Figure 2.9: Example of extraction rules in STALKER.
2.3 Extracting Semantic Information from Web Pages

There exist a few proposals that aim at extracting semantic information from web pages. Notice that although these systems are not directly related with the WebMeaning framework, it is interesting to describe some proposals from this field in order to have an overall view of existing web extractors.

2.3.1 Squeal

This proposal [Thesis and Spertus, 1998] was developed by Ellen Spertus. It aims at viewing the web as an enormous database, where structural relations are represented as database relations. This way, it provides an uniform framework for accessing the diverse types of web structure.

Implementing the abstraction of the web as a database is done through the Squeal ontology. It is represented by a SQL database schema, which serves as the user’s view of the web, allowing the user to focus on structural information without needing to be aware of the representation. Once in this representation, the Structured Query Language (SQL) can be used to have access to it. Figure 2.10 shows two SQL queries to extract information about the structure of the MIT web site.

In Squeal, we can inquire to get knowledge about the structure of a web page, and it is possible, because Squeal relates hypertext conventions to semantic relationships. Nevertheless, we are interested in relating the information residing in web pages with semantics.

2.3.2 WebOntEx

WebOntEx stands for Web Ontology Extraction. Its goal is to extract web ontologies semi-automatically by analyzing unstructured and semi-structured web pages that are in the same application domain, and to convert the ontology to XML DTD (Document Type Definition).

The main module of WebOntEx is the heuristic analyzer module that can be applied to any application domain. It uses inductive logic programming to classify the concepts into entity types, to get the relationships, attributes, and superclass/subclass hierarchies and to store this knowledge in a relational database. The DTD Creator module converts the information in the relational
2.3. Extracting Semantic Information from Web Pages

Query: How many times does each type of tag appear on www.mit.edu?

```
SELECT name, COUNT(*) FROM tag
WHERE url_id = url_id('www.mit.edu')
GROUP BY name;
```

Result:

<table>
<thead>
<tr>
<th>name</th>
<th>count</th>
</tr>
</thead>
<tbody>
<tr>
<td>!DOCTYPE</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>42</td>
</tr>
<tr>
<td>BODY</td>
<td>1</td>
</tr>
<tr>
<td>FNORD</td>
<td>15</td>
</tr>
<tr>
<td>HEAD</td>
<td>1</td>
</tr>
<tr>
<td>HTML</td>
<td>1</td>
</tr>
<tr>
<td>IMG</td>
<td>10</td>
</tr>
<tr>
<td>P</td>
<td>16</td>
</tr>
<tr>
<td>TITLE</td>
<td>1</td>
</tr>
</tbody>
</table>

Query: What tags appear within the first one hundred characters?

```
SELECT name, tag_id FROM tag
WHERE url_id = url_id('www.mit.edu')
AND startOffset < 100;
```

Result:

<table>
<thead>
<tr>
<th>name</th>
<th>tag_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>!DOCTYPE</td>
<td>1087</td>
</tr>
<tr>
<td>HTML</td>
<td>1088</td>
</tr>
<tr>
<td>HEAD</td>
<td>1089</td>
</tr>
<tr>
<td>TITLE</td>
<td>1090</td>
</tr>
</tbody>
</table>

Figure 2.10: Example of queries using Squeal.
database into an XML DTD; using a DTD to represent the knowledge limits using and sharing the extracted ontologies to applications that understand the XML tags defined by WebOntEx.

Figure 2.11 shows the DTD obtained using WebOntoEx to extract the semantics relations amongst the information of a company.

## 2.4 Extracting Knowledge from the Web

The proposals presented in this Section have the same goals that the Web-Meaning framework, that is, extracting knowledge from the web. The extracted knowledge adds value to software applications, because they can use it to accomplish new goals.

### 2.4.1 The Semantic Web

Some researchers and web developers are working on the Semantic Web. It was defined by Berners-Lee et al. [2001] as: “an extension to the current web in which information is given well-defined meaning, better enabling computers and people to work in cooperation”. The Semantic Web is not a separate Web but an extension to the current one. This extension is achieved by annotating web pages with meta-data that describe the concepts that define the semantics associated with the information in which we are interested. Ontologies play an important role in this task, and there are many ontological languages that aim at solving this problem, e.g., OWL, DAML+OIL [Horrocks et al., 2002], SHOE [Luke et al., 1997] or RDF-Schema [Brickley and Guha, 2000]. The Semantic Web shall simplify and improve the accuracy of current information extraction techniques tremendously. Nevertheless, this extension requires a great deal of effort to annotate current web pages with semantics.

Figure 2.12 shows the semantic web evolution as of the time of writing this document. The figures suggest that there are very little annotated pages available if we compare them with non-annotated pages. The DAML crawler\(^\text{12}\) reports 21,025 annotated web pages in march 2003, which is a negligible figure if we compare it with 2,110 millions, which is the estimated number of web pages (www.cyveillance.com). Current trends seem to suggest that.

\(^{12}\)www.daml.org/crawler
2.4. Extracting Knowledge from the Web

Figure 2.11: The XML DTD for a company schema..
the semantic web is not likely to be widely adopted in the forthcoming years [Hendler, 2001].

2.4.2 Omnibase and START

START and Omnibase are being developed at MIT’s Artificial Intelligence Laboratory [Katz and Lin, 2002]. START is a natural language question answering system and Omnibase is a virtual database that provides uniform access to Web resources. The authors argue that main goal of these systems is to develop a “smart reference librarian” that, although it might not be able to answer a question directly, it would be helpful because knows where to find the relevant knowledge.

START uses natural language annotations, they are sentences describing the contents of information segments (metadata that is machine-parseable). Annotations are used to describe the kinds of questions that some knowledge is able to answer. Figure 2.13 shows examples of queries, natural language annotations and responses using Omnibase and START.

The main drawbacks of this system are that the problem of knowledge extraction is limited to web pages with natural language annotations. It is
Sample web page:

The highest point on Mars is Olympus Mons, a huge shield volcano about 26 km (16 miles) high and 600 km (370 mi) across, about the same area as Arizona. Olympus Mons is also the largest known volcano in the solar system.

Example of annotation:

Mars’ highest point.
Largest volcano in the solar system.
Olympus Mons is very tall.

Example of questions:

What is the highest point on Mars?
Do you know how anything about Olympus Mons?
How tall is Olympus Mons?
Tell me how big Olympus Mons is.
What is the largest volcano in the solar system?

Figure 2.13: Example of queries in the START system.
also needed a lot of effort so that applications can talk to each other using natural language; therefore, the knowledge extracted is not appropriate to be used for software applications. Finally, it is a domain-dependent system.

2.4.3 The WEB→KB Project

The WEB→KB (World Wide Knowledge Base) [Craven et al., 2000] project at the University of Carnegie Mellon aims at developing a probabilistic, symbolic knowledge base that mirrors the contents of the web. It uses several machine learning algorithms for this task.

The WEB→KB takes two inputs, namely: an ontology that defines the concepts and relations of interest in order to create the knowledge base; a set of training data composed of web pages labelled with semantic information that represents instances of the ontology. With these inputs the system learns to extract information from similar pages and hyperlinks on the web.

The first task in this system is to identify new instances of ontology classes from web pages. This task is performed using two methods, the first method uses a statistical bag-of-words approach to classify web pages; in the second one, web pages are classified by an algorithm that learns first-order rules. The output of this system is the result of combining the predictions made by these classifiers.

An other important task is to learn to recognise relationships of interest that exist amongst extracted class instances. The hypothesis underlying the WEB→KB approach is that relationships among class instances are often represented by hyperlinks paths in the web. Thus, the solution is to induce rules that characterise the paths of the relation. This is done by means of a learning method that uses a first-order representation for its learned rules.

Finally, in some cases, the information to extract is not represented by web pages classification or hyperlinks amongst web pages, but by text embedded in web pages. To extract this information, the WEB→KB system uses the SRV wrapper presented in Section 2.2.1; therefore, this proposal presents the same drawbacks: it only generates single-slot rules and many data are needed to train the inductive algorithm.

Figure 2.14 shows an overview of WEB→KB system. The top shows an ontology that defines the meaning of the information to be extracted. Each partial box represents a class, and the arrows indicate specialisation relation-
2.5. Summary

2.5.1 Extracting Information from the Web

Induction wrappers are suited to extract information from the web, but they do not associate semantics with the data they extract, this being their major drawback. Therefore, from here on, we call current inductive wrappers syntactic because they extract syntactic information devoid of semantic formalisation that expresses its meaning. Another problem arises when we work with wrappers, the problem is due to the dynamism inherent to web sites. Wrappers are based on properties of web pages or characteristics of the information to extract. Changes to a web page can invalidate a wrapper. Therefore, it is necessary additional mechanisms to detect if the information extracted is valid, making it possible to extraction rules to be regenerated automatically if there are changes in the layout of the web page [Muslea et al., 1998, Kushmerick, 1999, 2000].

Table 2.1 summarises the characteristics of the inductive wrappers systems presented in this section. The first three columns indicate the type of web page that is handled, the fourth column indicates if the wrapper can perform multi-slot extraction; The last two columns indicate if system can handle missing items and permutations. All this characteristics helps to show the powerful of the wrappers systems studied.

Notice that the row corresponding to our proposal uses the symbol Θ to indicate its characteristics, it is because WebMeanig uses inductive wrappers to extract information. Thus, the powerful of WebMeaning depends on the wrapper used.

\[\text{This task is possible in some cases. If there are changes in the information sources, the wrapper cannot be regenerated automatically, and the user needs to train the inductive algorithm that builds the wrapper again.}\]
Chapter 2. Related work

Figure 2.14: Example of the WEB→KB system.
2.5. Summary

<table>
<thead>
<tr>
<th>Name</th>
<th>Struct</th>
<th>Semi</th>
<th>UnStruct</th>
<th>M-slot</th>
<th>Missing</th>
<th>Permuta</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rapier</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>SRV</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WHISK</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>WIEN</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>SoftMealy</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td></td>
<td>*</td>
</tr>
<tr>
<td>STALKER</td>
<td>*</td>
<td>*</td>
<td></td>
<td></td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>WebMeaning</td>
<td>θ</td>
<td>θ</td>
<td>θ</td>
<td>θ</td>
<td>θ</td>
<td>θ</td>
</tr>
</tbody>
</table>

Table 2.1: Summary of inductive wrappers systems.

2.5.2 Extracting Semantic Information from the Web

In WebMeaning the semantic information about data on a web page must be described by a human knowledge engineer. This is because the automatically extraction of concepts and relationships amongst them is a very difficult task. The accuracy of these systems is not enough to make that applications use their results. Thus, a lot of research is needed to solve the problem of semantic information extraction.

2.5.3 Extracting Knowledge from the Web

The semantic web will solve the problem of knowledge extraction, but current trends seem to suggest that it is not likely to be adopted in the forthcoming years. This argues for an automatic solution to extract semantically-meaningful information from today’s web in the meanwhile.

We think that the START system is not appropriate to solve the problem of knowledge extraction, because natural language is not understandable by software applications. However, the output from the START system can be useful to help humans.

Our work is most closely related to the WEB→KB system, in which the knowledge extraction process is focused on making available the knowledge on web to software applications. The main differences between WebMeaning and WEB→KB are as follows:
• The WEB→KB uses the SRV wrapper to extract information from the web; the WebMeaning framework, instead of is open to use any wrappers.

• The WebMeaning framework uses an algorithm to detect changes on web. WEB→KB does not use it. Therefore, WebMeaning improves adaptability to web changes.

• In WebMeaning, the access to the extracted knowledge is managed by core applications called knowledge channels (software agents). This way, we give a point of access to the knowledge base. In the WEB→KB system, applications query directly the knowledge base.

• WEB→KB can extract relations from hyperlinks. In WebMeaning the client is the responsible of getting these relations.

• In WEB→KB rules learned can be used in others web sites in a same application domain. In WebMeaning there is not a fix relation between the domain and the representation of the knowledge (Ontology) from this domain. For instance, two different web sites offering information about books may have different Ontologies to represent the knowledge.
Chapter 3

A Description of WebMeaning

A good scientist is a person with original ideas. A good engineer is a person who makes a design that works with as few original ideas as possible. There are no prima donnas in engineering.

Freeman Dyson, Disturbing the Universe, 1979.

This chapter presents the WebMeaning framework. It is organised to show the two components of WebMeaning. First, Section 3.1 describes a method to construct applications that extracts the knowledge from the web; Section 3.2 illustrates the reference architecture to have access to the knowledge extracted. Finally, Section 3.3 summarises the ideas in this chapter.

3.1 A Method to Construct Knowledge Channels

Figure 3.1 overviews the method defined in WebMeaning to construct knowledge channels. This figure sketches the tasks and the inputs and outputs expected.

Now, we describe each of these tasks. They will be specified formally in the next chapter.
Chapter 3. A Description of WebMeaning

Figure 3.1: Overview of the method to construct knowledge channels.
3.1.1 Generating the syntactic wrapper

The first task is to construct a syntactic wrapper to extract information from some web pages. WebMeaning supplies a set of implemented wrapper generators; the user only needs to select an appropriate wrapper for the web site in which the information resides. This selection is carried out by answering next questions:

- Is it needed single-slot or multi-slot extraction?
- Are web pages structured, semi-structured or unstructured?
- Do web pages contain slots with missing attributes, or attributes with multiple values, or attribute permutations?
- Do we need a quick or a robust wrapper?

For instance, regarding the web page in Figure 1.1, it is clear that we need a wrapper able to perform a multi-slot extraction, that works with structured web pages and it is not necessary that it deals with slots with missing attributes, or attributes with multiple values, or attribute permutations. For this web page, we could use the WHISK inductive wrapper system, because it fits all the requirements.

Once is selected the syntactic wrapper generator we need to feed it with web pages containing sample data. It outputs a set of extraction rules that we can use to set up an inductive generic wrapper. So far, we have a piece of software able to extract information from this web site.

3.1.2 Generating the Syntactic Verifier

In this task, we deal with web dynamism, the main goal is to get an algorithm that takes the information extracted by a wrapper as input and decides if the wrapper is working correctly, or, on the contrary, is it invalid because of changes in the web page structure. We call them syntactic verifiers because the decision about the wrapper’s being valid is based solely on syntactic properties of the information extracted.

In WebMeaning we use the Rapture syntactic verifier [Kushmerick, 2000]. It uses statistical features, such as length, number of words, number of special characters, or number of digits to characterise the information. It learns
the parameters of normal distributions describing the feature distributions of the extracted information. This values help to decide if the wrapper is valid by analysing the statistical values of the information extracted. Therefore, to carry out this task we need to train RAPIER with positive sample data, that is data extracted correctly by the wrapper.

3.1.3 Defining the Semantic Description

In order to extract knowledge from the web, we need to construct a semantic wrapper. It is composed of a syntactic wrapper and a semantic translator. The syntactic wrapper extracts the structured information from that web page, and the semantic translator assigns then meaning to it by means of an ontology. Figure 3.2 shows this idea.

The semantic translator needs the user specify a semantic description that relates the information to be extracted with the ontology that gives meaning to the information to be extracted. Thus, the semantic translator allows us to translate the information extracted by the syntactic wrapper to knowledge.

There are many formalisms to dealt with knowledge, namely: semantic networks [Quillian., 1967], frames systems [Minsky, 1975], logic, decision trees, and so on. Their aim is to represent ontologies, which are specifications of concepts and relationships amongst them in a concrete domain. We use a first–order logic language to represent ontologies and instances of ontologies.

\footnote{We hope that future versions of WebMeaning can extract automatically this semantic description.}
First-order languages (FOL) offer us the power and flexibility needed to describe knowledge. Many knowledge representation languages and structures can be formulated in first-order logic [Heflin, 2001]. Then, we are able to use a wide range of knowledge representation formalisms; we only need to define a mechanism to translate from a formalism to FOL and from FOL to this formalism.

We use the standard KIF (Knowledge Interchange Format) [Geneserith, 1991, Ginsberg, 1991] defined at the university of Stanford to represent and handle FOL in a computer understandable way. For instance, a semantic wrapper could output the following ground predicate terms in KIF from web page in Figure 1.1:

(Golfer Rich_Beem) (Golfer Chris_Riley)
(Score Rich_Beem 278) (Score Chris_Riley 283)
(Position Rich_Beem 1) (Position Chris_Riley 3)

(Golfer Tiger_Woods) (Golfer Justin_Leonard)
(Score Tiger_Woods 279) (Score Justin_Leonard 284)
(Position Tiger_Woods 2) (Position Justin_Leonard 4)
...

3.1.4 Specifying the Navigation to Web Pages

The access to the page that contains the information in which we are interested may involve navigating through a series of intermediate pages, e.g., login or index pages. For instance, suppose that we are interested in having access to the information about books that Amazon offers us; first, we need to go to the Amazon web page at www.amazon.com, then submit a form with keywords to locate the books in which we are interested; Amazon returns a set of web pages constructed dynamically with the results.

In order to simulate the navigation of a human user, WebMeaning incorporates a framework implemented in Java. It defines the navigation as a state machine, where a state references a set of web pages; and a transition from State 0 to State 1 references the process of navigating from pages that State 0 defines to pages that State 1 defines. Thus, the problem of web navigation is solved in WebMeaning by executing the machine that represents the navigation.

Form ap2 = new Form();
ap2.setName("searchform");
ap2.addValue("url", "index=books");
ap2.addValue("field-keywords", "Java");
Action a2 = new SubmitForm(ap2);

n.addAction(a2);

BuildPager BP = new BuildPager();
n.addAction(BP.NextImagePager(new Pattern("button-more-results.gif")));
n.run();

Programme 3.1: A piece of code representing a navigational state machine.

The output of this task is a Java function that gathers web pages. This function uses input parameters to specify the data necessary in the process of navigation, as the keywords used to fill forms. For instance, next piece of code represents a navigation state machine. It means the following: the navigation process begins at http://www.amazon.com, first transition is defined by submitting a form called searchform, filling previously the fields url and field-keywords with the values index=books and Java respectively (we are looking for a book about Java). Then, a pager that allows to gather all the results web pages returned by Amazon is defined; It uses a image button "button-more-results.gif" to access to the next results page.

3.2 The WebMeaning Reference Architecture

Figure 3.2 sketches the architecture of our proposal. As we mentioned above, knowledge channels are at its core agents because they specialise in extracting knowledge from web pages, and are able to react to information inquiries (reactivity) from other agents (social ability). They act in the back-
ground proactively according to a predefined schedule to extract knowledge.

There is also an agent broker for information extraction that acts as a trader between the agents that need information from the web and the set of available knowledge channels. When an agent needs some knowledge, it contacts the broker, which redirects the request to the appropriate information channel, if possible. This way, agents need not be aware of the existence of different KCs, which can thus be adapted, created or removed from the system transparently.

We use ACL [FIPA, 2000] as a transport language to send messages from an agent to another. The content of the messages describes how an agent needs to interact with another, and it is written in KIF. Figure 3.2 shows the brokering protocol [Finin et al., 1997] to communicate user agents with the KCs using the notation AUML [Group], 2001, Odell et al., 2000]. When an initiator agent sends a message with the performative proxy to the broker, it then replies with one of the following standard messages: not-understood, refuse or agree. If the broker agrees on the inquiry, it then searches for an adequate KC to serve it. If not found, it then sends a failure-no-match message to the initiator; otherwise, it tries to contact the KC and passes the inquiry onto it. If the broker succeeds in communicating with the KC, it shall later send the requested information to the initiator; otherwise, a failure-com-IC message is sent back to the initiator, which indicates that an appropriate KC exists, but cannot respond.
Figure 3.4: Broker Interaction Protocol in AUML.
3.3 Summary

In order to extract information from the web, WebMeaning uses inductive wrappers. Using inductive wrappers allows us to take advantage of all the work developed in this field, as boosted techniques or verification algorithms. Nevertheless, in this first version of WebMeaning we cannot represent directly hierarchical information (we cannot use the STALKER wrapper); overcoming this problem requires to construct several wrappers (one for each level in the hierarchy).

In previous chapter, we identified two major drawbacks that we need to address in order to get a reliable solution; the former was that syntactic wrappers do not associate semantics with the data extracted, we overcome this problem by extending them with an algorithm that gives semantics to the extracted information, we call these new wrappers semantic wrappers that uses a semantic description to translate the information into knowledge. The latter is that changes in a web page can invalidate a syntactic wrapper, this problem is overcome by using syntactic verification algorithms, they check the extracted information to insure that syntactic wrapper works correctly.

WebMeaning also provides a mechanism to define the navigation to the web pages that contains the information of interest. This mechanism is a framework that user can use and extend. This is interesting because in other proposals navigation is hidden to users.

The WebMeaning architecture has been presented as the appropriate mechanism to access the knowledge. It incorporates a broker agent that has information about registered knowledge channels (yellow pages). This way, the agents that need to use the extracted knowledge need not be aware of the existence of different KCs, which can thus be adapted, created or removed from the system transparently.

Finally, the best publications related with the main results of this chapter are the following ones:


Chapter 4

A Model for WebMeaning

"... all models are wrong; the practical question is how wrong do they have to be to not be useful."


This chapter presents knowledge channels formally. They are agents composed of semantic wrappers to extract knowledge, a navigation state machine to surf the web, and a semantic verifier to check the extracted knowledge. The Chapter is organised as follows: Sections 4.1 and 4.2 present specifications of concepts related with syntactic information extraction, namely: syntactic wrapper and syntactic verification; Section 4.3 extends these terms to deal with web knowledge; Section 4.4 shows the process of surfing the web as an execution of a navigation state machine (these ideas are brought together to define in Section 4.5 the knowledge channels formally); finally, Section 4.6 summarises the ideas in this chapter.

4.1 Syntactic Wrappers

WebMeaning is open to use any existing wrapper to extract information. To capture this idea we give a formal specification of a syntactic wrapper that allows adding new wrappers to WebMeaning.
**Definition 1** A syntactic wrapper is a function that takes a web page as input, and returns structured information.

Next schema specifies a syntactic wrapper:

\[
\begin{align*}
\text{Wrapper} &: \text{WebPage} \rightarrow \text{Information} \\
\text{dom} \text{Wrapper} &\neq \emptyset \\
\text{Datum} &= P \text{ String} \\
\text{Data} &= \text{seq Datum} \\
\text{Information} &= P \text{ Data}
\end{align*}
\]

A syntactic wrapper is modelled as a partial function because its domain is a subset of web pages that defines its scope, i.e., the web pages with which the wrapper can be used. It is is represented as the following set:

\[
\{ \forall p : \text{WebPage} \mid p \in \text{dom Wrapper} \bullet \text{Wrapper}(p) \}
\]

The output is modelled as data type Information, which is a set of Data; this allows us to deal with wrappers able to perform a multi-slot extraction. Data is sequence of Datum, which allows us to have a structured vision of the data to be extracted and to set a location for each datum (solving the problem of representing attribute permutations). Datum represents attributes to be extracted, and they are specified as a set of strings; this allows us to deal with missing attributes (the empty set), or attributes with multiple values (a set with multiple values).

**Example 1** If we were interested in extracting information about the position and score of golfers in a PGA championship, a syntactic wrapper (able to perform a multi-slot extraction) would output the following Information from the web page in Figure 1.1:

\[
\begin{align*}
\{(\text{Rich Beem}), \{278\}, \{1\}, \\
(\text{Tiger Woods}), \{279\}, \{2\}, \\
(\text{Chris Riley}), \{283\}, \{3\}, \\
(\text{Justin Leonard}), \{284\}, \{4\}, \\
\ldots\}
\end{align*}
\]

Syntactic wrappers take a web page as input, on the contrary wrappers uses a different format to represent the output. Thus, the user must define
a function that translates the output format of the syntactic wrapper to the one expected by WebMeaning, that is Information. We call this function Output Adapter.

**Example 2** The output format of some wrappers described in previous chapter is a template of the following form.

Golfer
  Name: Rich Beem
  Score: 278
  Position: 1

Golfer
  Name: Tiger Woods
  Score: 279
  Position: 2

... Translating this frame into an Information structure is a simple task. It involves the construction of a simple parser that performs a rendering between two simple formats.

# 4.2 Syntactic Verification

We specify a syntactic verifier as a predicate on the information extracted that ensures that it is satisfied if the wrapper works correctly.

\[
\text{SyntacticVerification} : \Lambda \times \text{Information}
\]

Type \(\Lambda\) references some parameters and if we give a concrete definition for \(\Lambda\), we would have a syntactic checker of wrappers. Thus, a syntactic wrapper is reliable if it satisfies next formula:

\[
\forall p : \text{WebPage} \mid p \in \text{dom Wrapper} \bullet \text{SyntacticVerification}(\lambda, \text{Wrapper}(p))
\]

Where \(\lambda : \Lambda\) is the information needed by the SyntacticVerification predicate.

As was mentioned in previous chapter, WebMeaning uses the Rapture syntactic verifier [Kushmerick, 2000] defined by Kushmerick, and \(\Lambda\) is defined...
as a set of real number that stores normal distributions of some measurements (html tag density, that is, number of \langle and \rangle symbols; also the length and number of words are measured) of the data extracted.

The syntactic wrapper specification can now be extended in order to deal with changes in the web. The idea is to use a syntactic verifier predicate on the information extracted, that assure that syntactic wrapper works correctly. This objective is achieved by the wrapper maintenance task, that consist on reconstructing the wrapper if the syntactic verifier predicate does not satisfy.

\[
\begin{align*}
\text{Wrapper} : & \text{WebPage} \rightarrow \text{Information} \\
\lambda : & \Lambda \\
\text{dom} \text{Wrapper} \neq \emptyset \\
\forall w : & \text{WebPage} \mid w \in \text{dom Wrapper} \cdot \text{SyntacticVerification}(\lambda, \text{Wrapper}(w))
\end{align*}
\]

4.3 Semantic Wrappers

4.3.1 Dealing with Knowledge

Some authors [Guarino, 1998, Heflin, 2001] have specified a formal model for ontologies; our formalisation builds on the work by Heflin in his PhD dissertation [Heflin, 2001].

Definition 2 Let \( \mathcal{L} \) be a logical language; an \textbf{ontology} is a tuple \( (P, A) \), where \( P \) is a subset of the vocabulary of predicate symbols of \( \mathcal{L} \) and \( A \) is a subset of well–formed formula in \( \mathcal{L} \) (axioms). Thus, an ontology is a subset of \( \mathcal{L} \) in which concepts are specified by predicates and relationships amongst then are specified as a set of axioms.

In Appendix A, we specify some concepts related to logical languages that establish the basis of our model. In our proposal, a logical language \( (\mathcal{L}) \) is characterized by a vocabulary of constant identifiers \( (\text{Ident}_c) \), a vocabulary of variable identifiers \( (\text{Ident}_v) \), a vocabulary of function identifiers \( (\text{Ident}_f) \), a vocabulary of predicate identifiers \( (\text{Ident}_p) \) and a (in)finite set of well–formed formula \( (\text{Wff}) \), which is a subset of the formula derived from \( \mathcal{L} \). For the sake of simplicity, we assume that \( \text{Ident}_f = \emptyset \).

Next schema specifies an ontology. Three constrains are imposed: the former states that \( P \) and \( A \) are non–empty subsets of the set of predicate symbols
4.3. Semantic Wrappers

and well-formed formula of \( \mathcal{L} \), respectively; the second, asserts that axioms are defined using the predicate symbols in \( P^{11} \); the latter asserts that the set of axioms is consistent. Predicate \( \vdash \) references a theorem prover; let be \( F : \mathbb{P} \text{Wff} \), and \( f : \text{Wff} \) then \( F \vdash f \) is satisfied if \( f \) is formally provable or derivable from \( F \), thus \( f \) belongs to the set of all Well-formed formula that we can obtain from \( F \) (theory of \( F \)).

Definition 3 An instance of a concept, specified in an ontology, is an interpretation of this concept over some domain. In information extraction this domain is established by the information to be extracted.

We model instances as ground predicate atoms. Thus, they are well-formed formula. We specify the set of all instances that we can derive from an ontology by the function \( \text{GroundPredicateAtoms} \):

\[
\text{GroundPredicateAtoms} : \text{Ontology} \rightarrow \mathbb{P} \text{Wff}
\]

\[
\forall o : \text{Ontology} \bullet \text{GroundPredicateAtoms}(o) = \{ f : \text{Wff}; \ ip : \text{Ident}_p; \ sc : \text{seq}_1 \text{Term}; \ ic : \text{Ident}_c \ |
\begin{align*}
& (f = \text{atom}(\text{pred}(ip, sc))) \land \\
& \forall c : \text{Term} \bullet c \in sc \bullet c = \text{const}(ic) \land \\
& \text{PredSyms}(f) \subseteq o.P \bullet f
\end{align*}
\]

Definition 4 A Knowledge Base (KB) is a tuple \( (O, K) \), where \( O \) is an ontology and \( K \) a set of instances of concepts specified in \( O \).

A KB is specified as follows:

\(^{11}\) The function \( \text{PredSyms} \) is specified in Appendix A. It returns the set of predicate symbols in a formula.
Chapter 4. A Model for WebMeaning

The constrains imposed assert that the instances are formed with predicates defined in the ontology and they are ground predicate atoms.

Example 3 The following object defines a KB in the domain of a golf championship in which we were interested in modelling knowledge about the position and score of golfers in a PGA championship (for the sake of readability, we do not use the abstract syntax in Appendix A. The mapping between this syntax and the usual logic symbols is straightforward):

\[
\begin{align*}
KB_0 = & \{ O \leadsto \{ \text{Person, Golfer, Score, Position} \}, \\
A \leadsto & \{ \forall x \bullet \text{Golfer}(x) \Rightarrow \text{Person}(x), \\
& \forall x \bullet \exists y \bullet \text{Golfer}(x) \Rightarrow \text{Score}(x, y), \\
& \forall x \bullet \exists y \bullet \text{Golfer}(x) \Rightarrow \text{Position}(x, y) \} , \\
K \leadsto & \{ \text{Golfer}(\text{Rich Beem}), \text{Score}(\text{Rich Beem, 278}), \\
& \text{Position}(\text{Rich Beem, 1}) \} \}
\end{align*}
\]

The ontology has four predicate symbols called Person, Golfer, Score and Position; the first axiom asserts that every Golfer is a Person; the second one states that every Golfer has a Score, where y represents the total number of points obtained; the last one asserts that every Golfer has a Position y in the championship. The instances in KB_0 can be interpreted using the ontology, and they asserts that Rich Beem is a golfer, and he is the first in the ranking with 278 points.

4.3.2 A formal Specification for Semantic Wrappers

A semantic wrapper is an extension to current syntactic wrappers.

Definition 5 A semantic wrapper is a function that takes a web page as input, and returns a set of instances of concepts defined in an ontology that represents the information of interest.
4.3. Semantic Wrappers

A semantic wrapper is specified as the result of applying a semantic translator to the output of the syntactic wrapper.

\[
\text{SemanticWrapper} : \text{WebPage} \rightarrow \mathbb{P} \text{Wff} \\
\forall p : \text{WebPage} \mid p \in \text{dom Wrapper} \cdot \\
\text{SemanticWrapper}(p) = \text{SemanticTranslator}(\text{Wrapper}(p))
\]

The semantic translator needs the user to specify a semantic description that relates the information to be extracted with the predicates defined in the ontology to perform this task.

**Definition 6** A **semantic description** (SD) is a representation of the relationships amongst the symbols of predicates from an ontology and the positions that their arguments occupy in an Information structure. Thus, each predicate \( P \) is associated with \( n \) natural numbers, where \( n \) is the arity of \( P \).

An SD is modelled using the following schema, which is composed of three elements: an ontology (\( O \)), a set of predicate symbols\(^{12} \) (\( S_p \)) and a function (\( \text{Pos} \)) that maps predicate symbols onto the location of a Datum in the Data that belongs to an Information structure. This scheme also asserts that \( S_p \) is a subset of the set of predicates symbols in \( O \), and the domain of \( \text{Pos} \) is a subset of the symbols in \( S_p \).

\[
\begin{align*}
\text{SemanticDescription} \\
O & : \text{Ontology} \\
S_p & : \mathbb{P} \text{Ident}_p \\
\text{Pos} & : \text{Ident}_p \rightarrow \text{seq}_1 \mathbb{N} \\
S_p & \subseteq O.P \\
\text{dom Pos} & = S_p
\end{align*}
\]

**Example 4** In our case study, we can define the following semantic description in which \( o_0 \) was presented in Example ??:

\(^{12} \) We might not need to use all of the predicates defined in an ontology to give meaning to the information extracted.
\[ O \leadsto o_0, \\
S_p \leadsto \{\text{Golfer, Score, Position}\}, \\
\text{Pos} \leadsto \{\text{Golfer} \mapsto \langle 1 \rangle, \\
\text{Score} \mapsto \langle 1, 2 \rangle, \\
\text{Position} \mapsto \langle 1, 3 \rangle\} \]

In this SD, predicate \text{Golfer} takes constant values from location \text{Pos} \text{(Golfer)} of each piece of Data in an Information structure, In this case, the first position of the sequence. Predicate \text{Score} takes its values from \text{Pos} \text{(Score)} = \langle 1, 2 \rangle^{13}, and so on. Thus, it is possible to generate automatically well-formed formulae that express the meaning of the information for all the Data elements in an Information structure extracted.

**Definition 7** A **semantic translator** is a function that receives the Information structure obtained using a syntactic wrapper as input and uses a semantic description specified by the user, and outputs a set of instances.

\[ sd : \text{SemanticDescription} \]

\[ \text{SemanticTranslator} : \text{Information} \rightarrow \mathbb{P} \text{Wff} \]

\[ \forall i : \text{Information} \mid i \in \text{ran Wrapper} \bullet \text{SemanticTranslator}(i) = \bigcup \{d : \text{Data} \mid d \in i \bullet \text{buildWffs}(d)\} \]

Function \text{buildWffs} returns the set of well-formed formulae for each piece of Data in an Information structure. It is defined as follows\(^{14}\)

\[ \text{buildWffs} : \text{Data} \rightarrow \mathbb{P} \text{Wff} \]

\[ \forall e : \text{Data}; t : \mathbb{P} (\text{Ident}_p \times \text{Data}) \mid e \in \bigcup \text{ranWrapper} \wedge t = \{x : sd.S_p \bullet (x, e \mid \{n : \text{ran Pos}(x) \bullet e(n)\})\} \bullet \text{buildWffs}(e) = \bigcup \{k : t \bullet \text{BuildPredicates}(k)\} \]

The function \text{BuildPredicates} is specified as follows:

---

\(^{13}\)The arguments in a predicate atom have a strict order. Using a sequence allows us to get them properly. For instance, if \text{Pos} \text{(Score)} were \langle 2, 1 \rangle, the result would be erroneous: \text{Score(278, Tiger.Woods)} states that the score of 278 is Tiger.Woods.

\(^{14}\)\(A \mid \nabla B\) filters the elements in sequence \(A\) that belong to set \(B\). For instance, if \(A = \{\text{sep, oct, nov, dec, jan, feb, mar, apr}\}\) and \(B = \langle \text{jun, nov, feb, jul} \rangle\), it follows that \(A \mid B = \langle \text{nov, feb} \rangle\).
BuildPredicates takes a pair composed of an identifier of a predicate and a piece of Data from an Information structure; it returns a set of predicate atoms of the form \( \text{atom}(\text{pred}(ip, si)) \) in which \( ip \) denotes a predicate symbol and \( si \) the sequence of constants to which it is applied.

**Example 5** The following instances represent the knowledge extracted by a semantic wrapper from the web page in Figure 1.1:

\[
\{\text{atom}(\text{pred}(\text{Golfer}, \langle\text{const}(\text{Rich} \cdots \text{Beem})\rangle)), \\
\text{atom}(\text{pred}(\text{Score}, \langle\text{const}(\text{Rich} \cdots \text{Beem}), \text{const}(278)\rangle)), \\
\text{atom}(\text{pred}(\text{Position}, \langle\text{const}(\text{Rich} \cdots \text{Beem}), \text{const}(1)\rangle)), \\
\text{atom}(\text{pred}(\text{Golfer}, \langle\text{const}(\text{Tiger} \cdots \text{Woods})\rangle)), \\
\text{atom}(\text{pred}(\text{Score}, \langle\text{const}(\text{Tiger} \cdots \text{Woods}), \text{const}(279)\rangle)), \\
\text{atom}(\text{pred}(\text{Position}, \langle\text{const}(\text{Tiger} \cdots \text{Woods}), \text{const}(2)\rangle)), \\
\text{atom}(\text{pred}(\text{Golfer}, \langle\text{const}(\text{Chris} \cdots \text{Riley})\rangle)), \\
\text{atom}(\text{pred}(\text{Score}, \langle\text{const}(\text{Chris} \cdots \text{Riley}), \text{const}(283)\rangle)), \\
\text{atom}(\text{pred}(\text{Position}, \langle\text{const}(\text{Chris} \cdots \text{Riley}), \text{const}(3)\rangle)), \\
\text{atom}(\text{pred}(\text{Golfer}, \langle\text{const}(\text{Justin} \cdots \text{Leonard})\rangle)), \\
\text{atom}(\text{pred}(\text{Score}, \langle\text{const}(\text{Justin} \cdots \text{Leonard}), \text{const}(284)\rangle)), \\
\text{atom}(\text{pred}(\text{Position}, \langle\text{const}(\text{Justin} \cdots \text{Leonard}), \text{const}(4)\rangle))\} 
\]

### 4.3.3 Semantic Verification

Semantic verification allows us to check the existing relations amongst the different concepts that check the semantics associated with the info extracted by a syntactic wrapper. Semantic verification detects inconsistencies in the information source used to feed a web site, whereas syntactic verification detects changes in the layout of a web site that invalidates the extraction process.

The solution to semantic errors is not to rebuild the syntactic wrapper since it works well, but to wait for the information to be corrected or to look for another site that offers the same information.

The semantic verification function is specified as follows:
Chapter 4. A Model for WebMeaning

Semantic verification is defined as a predicate. It is satisfied if the knowledge extracted is consistent with the axioms in the ontology under consideration. That is, it does not exist a formula \( f \) in the knowledge extracted such that \( f \) and \( \neg f \) belong to the theory obtained from the axioms in ontology and the formula that represent the instances.

4.4 Navigating through the Web

In order to simulate the navigation of a human user, we have developed a navigation framework. It defines navigation as a navigational state machine.

4.4.1 A Case Study

We illustrate our proposal by means of a simple example in which we are interested in having access to the information about books that Amazon offers us\textsuperscript{15}.

4.4.2 Navigational State Machine

Definition 8 A navigational state machine is a tuple \( M = (q_0, T, F) \) where \( q_0 \in \mathbb{P} \) WebPage, is the starting state; \( T \) is a sequence of transitions, and \( F \) is a subset that represents transitions that take us to web pages which contain the information in which we are interested.

A navigational state machine is formally specified as the following schema:

\[
[\text{Transition}, \text{TransitionParams}]
\]

\textsuperscript{15} This information is available at www.amazon.com
Two constrains are imposed in the schema: the former states that the starting state is non-empty subset of web pages; the latter asserts that the \( F \) set is composed by references to transitions defined on \( T \).

Transitions are functions that represent state changes; their profile is as follows: \( \text{Transition} : \mathbb{P} \text{WebPage} \times \text{TransitionParams} \rightarrow \mathbb{P} \text{WebPage} \); we need some parameters specified as the free type \( \text{TransitionParams} \) in order to execute them. According to previous specifications, transitions amongst states are defined as call to functions. We have defined a set of transitions that users can use to specify the navigation. They are classified in simple transitions, namely: FollowLink or SubmitForm. And complex transitions, that are special transitions that normally appears when we need to have access to a web page that offers information from a data base (dynamic web pages) namely: pagers and indexers.

The process of navigating the web automatically is simulated by an execution of the machine. An execution is the application of all the transitions defined on \( T \), that is the machine starts from \( q_0 \) and applies transitions from \( T \) until the last element of \( T \) is reached. The output of an execution is a set of web pages that contains the web pages reached from transitions referenced in \( F \). Therefore, an execution is defined as a function with the profile: \( \text{execution} : M \rightarrow \mathbb{P} \text{WebPage} \) where \( M \) is a navigational machine. It is specified axiomatically as a recursive function.

\[
\begin{align*}
\text{NavigationalStateMachine} \\
q_0 & : \mathbb{P} \text{WebPage} \\
T & : \text{seq}(\text{Transition} \times \text{TransitionParams}) \\
F & : \mathbb{P}(\text{Transition} \times \text{TransitionParams}) \\
q_0 & \neq \emptyset \\
F & \subseteq \text{dom} T
\end{align*}
\]
The base case is a navigational state machine in which the sequence of transitions $T$ has not any elements; when this case is reached an empty set of web pages is returned. The recursive call applies the first transition in $T$ and updates the starting state; if the transition applied is referenced in $F$, the web pages reached are then added to the output.

**Example 6** In order to gather the book with title *Core Java 2, Volume I: Fundamentals (6th Edition)* from Amazon, we need to go to the Amazon web page at www.amazon.com, then submit a form with keywords to locate the book in which we are interested; Amazon then, returns the web page with information about this book. This can be specified as follows:

$$
\begin{align*}
\text{AmazonNav} &= \{q_0 \sim \{\text{www.amazon.com}\} \\
T \sim & \langle(\text{SubmitForm, SubmitFormPar("Core Java 2, Volume I: Fundamentals (6th Edition")}) \\
K \sim & \{(\text{SubmitForm, SubmitFormPar("Core Java 2, Volume I: Fundamentals (6th Edition")}) \} \}
\end{align*}
$$

The AmazonNavigation machine is composed of one transition that submits the form in web page www.amazon.com with the keyword: Core Java 2, Volume I: Fundamentals (6th Edition). The web page that we reach when form is submitted is the output of execution(AmazonNav).

### 4.4.3 Pagers and Indexers

In order to present pagers and indexers, we present a slightly modified version of the previous case study in which we are interested in extracting information about the set of books written by a given author. If we write the name of an author in the above-mentioned search form, we can get a result page similar to the one presented in Figure 4.1. In this case, the initial page does not lead directly to the requested information. Instead, it leads to an index page in which we can find some links that point to pages that describe books written by that author, and a pager that allows to search for more results.
4.4. Navigating through the Web

Links to the pages that contains the information about books

Pager to iterate thorough the results

Figure 4.1: An example of a web page containing an iterator and a pager.
Pagers

We have identified two different pagers, the former uses some link or image labelled with *More results* to reference the next results page, for instance, figure 4.1 shows a pager of this characteristics; the latter enumerates the first $n$ web pages of results, thus, it indexes next $n$ results. For instance, a pager as $1, 2, 3, 4, 5, 6-10$ allows to access directly the first fives page of results, then in order to access the next five pages, we need to click on link $6-10$.

We deal with the first kind of pagers with a pair of transitions called *PagerMore* and *PagerMoreI*. The former, takes a regular expression pattern as input and looks for a piece of text describing a hyperlink inside the web page that matches the pattern; this process continues until no pattern was found in a result page; all pages referenced by links matching the pattern are stored. The latter, limit pattern matching to links with images; it look for image names that match the pattern. For instance, we use a *PagerMoreI* pager defined with the regular expression pattern "button-more-results.gif" to deal with the Amazon pager.

In order to deal with the second kind of pagers we have also implemented a pair of transition called *PagerIndex* and *PagerIndexI*. The former, takes two regular expression patterns as input and looks for pieces of text describing hyperlinks inside the web page that match the first pattern; the second pattern is needed to access next web page of results; this process continues until the second pattern is not found; all pages referenced by links matching the first pattern are stored. The latter, limit pattern matching to links with images; it look for image names that match the patterns. For instance, to deal with the example pager $1, 2, 3, 4, 5, 6-10$ we can define a *PagerIndex* pager defined with the regular expression patterns $[0-9]+$ to reference pages of results and $[0-9]+'-'[0-9]+$ to reference the following five pages of results.

Indexers

In order to deal with indexers we have developed an inductive wrapper system that we call *IWrappers*. This system must be trained with sample data (web pages and the urls corresponding to the links in which we are interested); with these information, an extraction rule is generated by means of an inductive algorithm. This rule sets up a generic algorithm to extract information from similar pages automatically.

The idea inside the Iwrappers system is that related links in a web page follow a common pattern in the tree DOM structure [Wood et al., 1998, Hors
et al., 2000, Nicol, 1998] of the web page. This hypothesis seems to be correct when web pages are generated dynamically in the server, but it is not easy when they are hand–generated by human users. However, when creating web pages, maintaining the structure to present data is a desirable principle that users follow.

The extraction rules generated by IWrappers are path patterns defined in the DOM tree. For instance, in order to extract links to web pages with information from a query in Google, IWrappers generates next extraction rule:

```
[ [ 0, a, 0, p, 1, blockquote, *, div, *, body, *, html, *
, #document]
, [0, a, *, p, *, div, *, body, *, html, *, #document] ]
```

The extraction rule define paths as a sequence of nodes in the tree document, these path goes from the root node of the html document (the element type #document) to the link we are interested to extract (the element type anchor a) and because a parent might have several childs of the same element type we need also to indicate the child or an asterisk to indicate that it does not matter the number of child. In previous example, the rule is composed of two paths, they indicate that links to be extracted are the ones that:

1. The first (0 indicates the first child) anchor (type of element a) that has as parent an element of type p (paragraph), being p the first child of an element of type blockquote, this blockquote is the second child of a parent of type div, and this a parent of type body, and this a parent a html element, finally the element of type html has as parent the root of the DOM tree.

2. The first anchor that has as parent an element of type p (paragraph), being p the first child of an element of type div, and this a parent of type body, and this a parent a html element, finally the element of type html has as parent the root of the DOM tree.

In order to construct an extraction rule, the inductive algorithm consists of identifying relative paths inside the DOM tree of the web page to the links of interest. This paths are relative because they establish a common parent node for links in sample data, then it calculates the paths in the tree from this

*Notice that these paths patterns specify links into the web page. That is, we can follow them to locate and extract a set of links.*
common parent to each link, and adds the paths from the common parent to the root of the DOM tree (in this piece of path added it does not matter of child); the algorithm then, merges these paths to generalise path patterns. This process is applied until all sample data can be explained with the rule generated.

The merging task consist of identify paths with identical sequences of elements and group them into a unique pattern path. The type of child elements are kept but the positions that child occupy are updated as follows: If two childs have different positions, then we update the position of this child with an asterisk, or else if they have the same position then the positions is maintained.

**Example 7** When the Iwrappers system takes the first web page from the sample data, it generates the following pattern paths for each link:

```
[0, a, 0, p, *, div, *, body, *, html, *, #document],
[0, a, 1, p, *, div, *, body, *, html, *, #document],
[0, a, 2, p, *, div, *, body, *, html, *, #document],
[0, a, 3, p, *, div, *, body, *, html, *, #document],
[0, a, 4, p, *, div, *, body, *, html, *, #document],
[0, a, 5, p, *, div, *, body, *, html, *, #document],
[0, a, 6, p, *, div, *, body, *, html, *, #document],
[0, a, 7, p, *, div, *, body, *, html, *, #document],
[0, a, 8, p, *, div, *, body, *, html, *, #document],
[0, a, 9, p, *, div, *, body, *, html, *, #document]
```

Then it merges this paths and get the following path pattern:

```
[[0, a, *, p, *, div, *, body, *, html, *, #document]]
```

Then, it tries to extract links from the web page using this rule and compares them with the ones in sample data. In this case the rule generated can extract the correct links, but it needs to check what happens with the rest of sample web pages in sample data.

It takes the following web page and tries to extract links, but links extracted do not correspond to the correct ones. This situation is illustrated in Figure 4.2; notice that the second link does not maintain the structure of first link.
Again, pattern paths to links are generated from a DOM tree, and they are merged with the one obtained in the previous merging process. The rules to merge are as follows:

\[
\begin{align*}
&[*, \text{a, }, *, \text{p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 0, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 1, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 2, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 0, p, 3, blockquote, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 4, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 5, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 6, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[1, \text{a, 7, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 8, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[0, \text{a, 9, p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}]
\end{align*}
\]

The result of merging this paths is a rule composed of two path pattern as follows:

\[
\begin{align*}
&[0, \text{a, 0, p, 3, blockquote, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}],
&[*, \text{a, }, *, \text{p, }, *, \text{div, }, *, \text{body, }, *, \text{html, }, *, \#\text{document}]
\end{align*}
\]

Now, links are correctly extracted from all web pages in sample data. Therefore the algorithm finishes.

### 4.5 Bringing all together: Knowledge Channels

A KC is responsible for managing a local knowledge base (KB). This knowledge is extracted from a web site using semantic wrappers. Before storing the knowledge in the KB, it is verified using semantic verification to check the existing relations amongst the different concepts that give semantics to the information extracted. Thus, a KC can answer queries from other agents that need some knowledge to accomplish its goals.

The schema bellow formalises a KC. It has a declarative part containing three variables; the former (SW) references the semantic wrapper to be used;
Figure 4.2: An Example of Doom tree for a web page from Amazon.
the second (NSM) is a navigational state machine; latter (SV) the semantic verifier.

```
KnowledgeChannel
SW : SemanticWrapper
SV : SemanticVerificator
NSM : NavigationalStateMachine
```

Next schema defines the overall state of our system. It is composed of a knowledge channel, a local KB and its environment. The environment (the perceivable features of a KC agent) is specified as set of web pages obtained in the navigation process, and we constraint that the semantic wrapper must be defined for these web pages.

```
KnowledgeChannelState
KC : KnowledgeChannel
KB : KnowledgeBase
Environment : \{WebPage\}

Environment = execution(KC.NSM)
Environment \subseteq \text{dom} KC.SW
```

The motivation of the knowledge channel is to synchronise the knowledge that resides in the knowledge base (KB) with the one in web pages. This motivation allows us define a KC as an autonomous piece of software. Next schema specifies the agent motivation. \(\Delta\) means that the state of the can change and the constrains indicate that all the knowledge in local KB must be on environment.

```
KnowledgeChannelMotivation

\Delta KnowledgeChannelState

\forall f : \text{Wff} \mid f \in KB.K \bullet \exists p : Environment \bullet f \in SW(p)
```

In order to address the synchronisation problem we have three possibilities:
1. To update the local KB, when an agent requests for knowledge. The advantage of this solution is that the knowledge extracted is sound with the one residing on web page. The main disadvantage is that if there exist many knowledge requests the knowledge channel can go down.

2. Periodically update the local KB. The advantage of this solution is that it is efficient because the number of web access depends of a predefined time period. The disadvantage is that the knowledge extracted may be not sound with the one residing on web page.

3. To use an event model in which an external system notifies to the knowledge channel about changes on information residing in web pages. The main advantage is that access to web page is delegated to other system. The disadvantage is that the external system can be the new bottle neck.

In this first version of WebMeaning we have select the second option. It is specified using a predicate $\text{timer}$, if it satisfied then a subset of previously not updated formula are updated.

$$\Delta \text{KnowledgeChannelState}$$

$$\text{LastKnowledgeUpdated} : \mathbb{P} \text{Wff}$$

$$\text{Timer} : t$$

$$(t \land (\exists sf : \mathbb{P} \text{Wff} \mid (sf \neq \varnothing \land sf \subseteq KB.K \land sf \cap \text{LastKnowledgeUpdated} = \varnothing) \bullet \exists p : \text{Environment} ; f : sf \bullet f \in SW(p)$$

$$\text{LastKnowledgeUpdated}' = sf$$

A KC serves requests from other agents, thus they show social ability. Delegating the task of knowledge extraction to KCs allows agent developers to achieve a complete separation between the knowledge extraction procedure and the logic or base functionality an agent encapsulates. An agent can send messages to a KC in order to extract knowledge. Knowledge requests are expressed as predicate symbols. The reply from the KC are ground predicate atoms that satisfy the predicates in a query or an empty set of formulae if the KB is erroneous.

Next schema formalises these ideas as a Z operation. $\Xi$ indicates that the state of the system is not modified at all, $Q?$ is the input query to the operation, and $R!$ denotes its result.
Example 8 If a KC receives query \( Q = \{ \text{Golfer} \} \), it should then reply:

\[
R = \{ \text{Golfer}(\text{Rich\_Beem}), \text{Golfer}(\text{Tiger\_Woods}), \text{Golfer}(\text{Chris\_Riley}), \ldots \}
\]

This knowledge can be used to infer new knowledge. It also makes it possible to reuse knowledge. For instance, an agent using the golfer ontology can infer that the golfers Rich Beem and Tiger Woods are people, according to the axiom \( \forall x \cdot \text{Golfer}(x) \Rightarrow \text{Person}(x) \), these knowledge can be shared by applications in order to collaborate to accomplish a task. For instance, we are developing an auction portal for a bookstore. User agents can move to a FIPA [FIPA, 2000] compliant agent platform installed in the portal to participate in auctions. When an agent arrives to the portal, it queries an auction manager agent to know the books in sale. Also the agent can use the knowledge about books extracted by a KC from www.Amazon.com.

4.6 Summary

In this chapter we have presented knowledge channels formally. They are agents composed of semantic wrappers to extract knowledge, a navigation state machine to surf the web, and a semantic verifier to check the extracted knowledge.

First, the syntactic wrappers and the syntactic verifications algorithm were specified, then we extended these terms to deal with web knowledge (semantic wrappers and semantic verifications algorithms). then, the process of surfing the web was modeled as an execution of a navigation state machine.
Finally, the best publications related with the main results of this chapter are the following ones:


Chapter 5

Conclusions and Future Work

People do not like to think. If one thinks, one must reach conclusions. Conclusions are not always pleasant.

Helen Keller (1880 - 1968).

In this chapter we conclude this pre–dissertation by summarizing our contributions and discussing directions for future work. It is organised as follows: Section 5.1 summarises our main conclusions; Section 5.2 discusses future research directions. Finally, Section 5.3 reports on three case studies in which the WebMeaning framework is valuable.

5.1 Conclusions

The goal of this pre–PHD was to support the idea that information residing on web pages is not understood by software agents. In the body of this pre–PHD, we presented strong motivation for this idea, supporting it with an example of a web site offering information about scores in a golf championship and showing the problems that appear when an agent is willing to retrieve the knowledge on web. These problem were due to the fact that the current web is mostly user–oriented.
Chapter 5. Conclusions and Future Work

The more interesting proposal presented in related work is the semantic web. It shall help extract information with well-defined semantics, regardless of the way it is rendered, but it does not seem it is going to be adopted in the immediate future, which argues for another solution to the problem in the meanwhile.

WebMeaning is our approach to extract semantically-meaningful information from today’s non-semantic. It achieves to delegate the knowledge extraction procedure to specialised knowledge channels agents, easing software development and promoting software reuse and maintainability. The core of WebMeaning are the knowledge channels; they are agents able to react to knowledge inquiries (reactivity) from other agents (social ability), and act in the background (autonomy) to maintain a local knowledge base (KB) with knowledge extracted from a web site (proactivity). In order to allow for semantic interoperability, the knowledge they manage references a number of concepts in a given application domain that are described by means of ontologies. Furthermore, they surf the web to access to the web pages that contains the knowledge.

5.2 Future Work

Future work might focus on the following aspects:

First, we think that description logics [Horrocks, 2002] is a good mechanism to represent knowledge. We have begun to evaluate the possibility of using description logics and we have presented it at Caise’03 (Conference on Advanced Information Systems Engineering), the referees’ reports makes that this idea was promising.

Description logics allows to specify a terminological hierarchy using a restricted set of first order formula (monadic/dyadic Predicate Logic statements with three variables). They usually are decidable and tractable using inference engines as RACER [Haarslev and Möller, 2001], but the inference services are restricted to classification and subsumption. That means, given formulae describing a classes, the classifier associated with a certain description logic will place them inside a hierarchy, and given an instance description, the classifier will determine the most specific classes to which the particular instance belongs.

Note that using first order logic for specifying axioms we gain expresivi-
ness, but FOL is semi-decidable and doing inferencing is computationally not tractable for large amounts of data and axioms.

Additionally, we are working in generating automatically the semantic descriptions that relates the information to be extracted with the ontology that gives meaning to this information. The solution is based on feeding the syntactic wrappers with web pages annotated semantically, and to infer the relationships with extracted information.

Finally, we feel that the work in this pre–PHD provides new opportunities for further research into various aspects. The framework could be extending to access to heterogeneous information sources as federate databases or knowledge bases. Accessing to database implies to translate the relational model into an ontology, but in a federated environment could be many ontologies for the same domain. In order to achieve semantic interoperability, these different ontologies must be able to interoperate or integrate with each other; this requires one ontology to map its vocabulary to the vocabulary in the other ontologies [Goñi et al., 1998, Wiederhold, 1997, Heflin and Hendler, 2000]. Accessing to knowledge bases implies to define translation schemes between ontologies, and the same problem appears, that is semantic interoperability.

5.3 Practical Validation of WebMeaning

In order to validate the WebMeaning framework, we are working in implementing three cases study from three different fields. They are:

- Intelligent agents community.
- Adaptative hypermedia community.
- Knowledge engineering / data base community.

5.3.1 Helping a Spanish Political Party to Attract Voters

In Corchuelo et al. [2002], we present a case study in which a web agent helps a Spanish political party decide if it should organise an outdoors meeting to attract voters.

Given that every political party wants to win the elections in which it participates, the decisions made during electoral campaigns are critical. Parties
are well aware that outdoor meetings may help achieve this goal, and they also know that it is difficult to carry through the plans when a bad decision is made. Thus, parties are interested in inferring adequate decisions from background knowledge to get as much votes for as possible.

With regard to Spain, companies such as Demoscopia\(^{11}\) or Sigma2\(^{12}\), as well as the National Statistics Institute\(^{13}\), have databases that record thousands of data about how voters behaved in past elections. They usually conduct polls and inquiry voters about several issues that are subliminally related to the party they are going to vote for. Such databases are usually sold to companies or parties so that they can infer knowledge from them. This knowledge may be represented, for instance, as a set of hierarchical decision rules [Riquelme et al., 2000] that may help forecast which party is more likely to win current elections in a city if they are fed with current data extracted from a new, fresh poll. Having these data in real–time is very important because the figures are changing constantly during a campaign.

By using this information, a party can reach decisions about which the best city to organise an outdoors meeting is. However, this decision cannot be reached without taking the weather into account. Organising an outdoors meeting is effective as long as the weather conditions are good; otherwise it might turn into a complete disaster.

Our goal is to design an agent able to help a political party decide the city where an outdoors meeting should be organised in order to increase the number of votes for. Such an agent needs to use data from on–line polls and on–line weather servers, and use them to feed a set of rules extracted from past election databases.

### 5.3.2 Web Site Personalisation

The Adaptative Hypermedia Community can use the WebMeaning to personalise web sites based on users’ profiles. In this respect, the information from the web that is interesting to a user can be extracted automatically and displayed in a suitable form.

We illustrate this idea in Arjona et al. [2002a], where we present a fictitious case study whose aim was the personalisation of an e-commerce portal that

\(^{11}\)www.demoscopia.com
\(^{12}\)www.sigmados.com
\(^{13}\)www.ine.es
sells VHS, DVDs, and so on. Using the information inferred from the buys by some customers, we can identify who the preferred actors are. Once the actors have been identified, we can develop an agent that uses the proposed framework. It queries an knowledge channel that is able to extract relevant information about this actor (birth name, location, filmography, ...) from several web sites, for example The Internet Movie Database (IMDb)\(^\text{14}\).

### 5.3.3 Knowledge Base about Web Services

There exist several web sites that offer information about web services, for instance salcentral.com. In this study case we aims to maintain a knowledge base about web services. Any application can query a knowledge channel to get the description of a service.

The Knowledge channel is the responsible of integrating different web information sources. It gives a uniform way to access to the knowledge it manages. Also, it automatically updates the information in the KB, removing old web services, adding new web services, and updating the web services that have changed.

\(^{14}\text{http://www.imdb.com/}\)
Appendix A

Well–formed formula

Let $Ident_c$, $Ident_v$, $Ident_f$, $Ident_p$ be the sets of identifiers of constants, variables, functions and predicates, respectively, in a first–order logical language. Then the complete language can be specified as the $\mathcal{Z}$ free type $Formula$ in the following way:

$$[Ident_c, Ident_v, Ident_f, Ident_p]$$

$$Term ::= \text{const}(Ident_c)$$
$$\mid \text{var}(Ident_v)$$
$$\mid \text{func}(Ident_f \times \text{seq}_1\ Term)$$

$$Atom ::= \text{pred}(Ident_p \times \text{seq}_1\ Term)$$
$$\mid \text{not}(Atom)$$
$$\mid \text{and}(Atom \times Atom)$$
$$\mid \text{or}(Atom \times Atom)$$
$$\mid \text{implies}(Atom \times Atom)$$
$$\mid \text{iff}(Atom \times Atom)$$

$$Formula ::= \text{atom}(Atom)$$
$$\mid \text{forall}(Ident_v \times Formula)$$
$$\mid \text{exists}(Ident_v \times Formula)$$
Appendix A. Well–formed formula

This states that a Formula is either an atom or an universal quantifier over a formula or an existencial quantifier over a formula. An atom is either an n-ary predicate or the negation of an atom or the conjunction of two atoms or the disjunction of two atoms or the implication formed from two atoms or the bi–implication formed from two atoms. A term is an identifier of constant or an identifier of variable or a n-ary function. To illustrate the use of this free type, formula $\forall P(x) \Rightarrow Q(x)$ is represented by the following term:

\[
\text{forall}(x, \text{atom}(\text{implies}(\text{pred}(P, \langle var(x) \rangle), \text{pred}(Q, \langle var(x) \rangle))))
\]

A well–formed formula (Wff) is a formula that does not contain any free variables, that is, its variables are bounded by universal or existencial quantifiers. In order to define the set of the well–formed formula in a logical language, we need to specify axiomatically a recursive function called FreeVars. It obtains the free variables in a formula or atom or term.

\[
\text{FormulaAtomTerm} ::= \text{Formula} \mid \text{Atom} \mid \text{Term}
\]

\[
\begin{align*}
\text{FreeVars} : \text{FormulaAtomTerm} & \rightarrow \mathbb{P} \text{Ident}_v \\
\forall f : \text{Formula}; a, a_1, a_2 : \text{Atom}; \ iv : \text{Ident}_v; \ ip : \text{Ident}_p; \ if : \text{Ident}_f & \bullet \\
\text{FreeVars}(\text{atom}(a)) &= \text{FreeVars}(a) \land \\
\text{FreeVars}(\text{forall}(iv, f)) &= \text{FreeVars}(f) \setminus \{iv\} \land \\
\text{FreeVars}(\text{exists}(iv, f)) &= \text{FreeVars}(f) \setminus \{iv\} \land \\
\text{FreeVars}(\text{not}(a)) &= \text{FreeVars}(a) \land \\
\text{FreeVars}(\text{and}(a_1, a_2)) &= \text{FreeVars}(a_1) \cup \text{FreeVars}(a_2) \land \\
\text{FreeVars}(\text{or}(a_1, a_2)) &= \text{FreeVars}(a_1) \cup \text{FreeVars}(a_2) \land \\
\text{FreeVars}(\text{implies}(a_1, a_2)) &= \text{FreeVars}(a_1) \cup \text{FreeVars}(a_2) \land \\
\text{FreeVars}(\text{iff}(a_1, a_2)) &= \text{FreeVars}(a_1) \cup \text{FreeVars}(a_2) \land \\
\text{FreeVars}(\text{pred}(ip, st)) &= \bigcup \{t : \text{Term} \mid t \in st \bullet \text{FreeVars}(t)\} \land \\
\text{FreeVars}(\text{var}(iv)) &= \{iv\} \land \\
\text{FreeVars}(\text{const}(c)) &= \emptyset \land \\
\text{FreeVars}(\text{func}(if, st)) &= \bigcup \{t : \text{Term} \mid t \in st \bullet \text{FreeVars}(t)\}
\end{align*}
\]

Set Wff is specified as the set of logical formula that does not have any free variables.
We can also obtain the set of predicate symbols in a formula:

\[
\text{PredSyms} : \text{FormulaAtom} \rightarrow \mathbb{P} \text{Ident}_p
\]

\[
\forall f : \text{Formula}; a, a_1, a_2 : \text{Atom}; iv : \text{Ident}_v; ip : \text{Ident}_p \cdot
\]

\[
\begin{align*}
\text{PredSyms}(\text{formula}(a)) &= \text{PredSyms}(a) \\
\text{PredSyms}(\forall(iv, f)) &= \text{PredSyms}(f) \\
\text{PredSyms}(\exists(iv, f)) &= \text{PredSyms}(f) \\
\text{PredSyms}(\text{not}(a)) &= \text{PredSyms}(a) \\
\text{PredSyms}(\text{and}(a_1, a_2)) &= \text{PredSyms}(a_1) \cup \text{PredSyms}(a_2) \\
\text{PredSyms}(\text{or}(a_1, a_2)) &= \text{PredSyms}(a_1) \cup \text{PredSyms}(a_2) \\
\text{PredSyms}(\text{implies}(a_1, a_2)) &= \text{PredSyms}(a_1) \cup \text{PredSyms}(a_2) \\
\text{PredSyms}(\text{iff}(a_1, a_2)) &= \text{PredSyms}(a_1) \cup \text{PredSyms}(a_2) \\
\text{PredSyms}(\text{pred}(ip, st)) &= \{ip\}
\end{align*}
\]
Appendix A. Well-formed formula
Appendix B

Curriculum

B.1 Education

Master Degrees in computer science. University of Seville (1997).

B.2 Professional Experience

From 1997 to 2000, I worked as a software engineer and a software consultant for Spanish companies such as Isotrol, S.A. and Norsistemas, S.A., where I got involved in several projects on electrical network modeling, digital TV platform, and e-commerce solutions.

Currently, I am a lecturer professor of computer science in the University of Seville. Where I have been teaching the following matters: Language Processors I, Language Processors II, and Formal Languages and Automata.
B.3 Publications

B.3.1 Journals

- **Title:** A Framework for Extracting Information with Semantics from the Web. An Application to Knowledge Discovery for web Agents  
  **Authors:** R. Corchuelo, J. Aguilar, and J. L. Arjona  
  **Reference:** The International Journal of Computers, Systems and Signals (IJCSS). ISSN 1608-5655.  
  **Volume:** 3, No. 2, 2002  
  **Initial page:** 12  
  **Final page:** 28  
  **Date:** Oct-2002

- **Title:** Extracción Automática de Información con Semántica de la Web  
  **Authors:** R. Corchuelo, J. L. Arjona, and A. Ruiz  
  **Reference:** NOVATICA. ISSN: 0211-2124; CODEN NOVAEC  
  **Volume:** 157  
  **Initial page:** 27  
  **Final page:** 33  
  **Date:** Jun-2002

- **Title:** Automatic Extraction of Semantically-Meaningful Information from the Web  
  **Authors:** R. Corchuelo, J. L. Arjona, and A. Ruiz  
  **Reference:** UPGRADE (The European Online Magazine for the Information Technology Professional)  
  **Volume:** Vol. III, No. 3  
  **Initial page:** 44  
  **Final page:** 51  
  **Date:** Jun-2002

- **Title:** Extracting Information from the Web  
  **Authors:** J. L. Arjona, R. Corchuelo, D. Ruiz, and M. Toro  
  **Reference:** Journal of Internet Technology (JIT). ISSN 1607-9264  
  **Volume:** 3, No. 4  
  **Date:** Oct-2002
B.3.2 Book Chapters

- **Title:** A Distributed Solution to Synchronous Multiparty Interaction  
  **Authors:** R. Corchuelo, D. Ruiz, M. Toro, J.L. Arjona, and J.M. Prieto  
  **Reference:** Recent Advances in Signal Processing and Communications. ISBN 960-8052-03-3  
  **Initial page:** 318  
  **Final page:** 323  
  **Date:** Ag-1999  
  **Editorial:** World Scientific Engineering Society. Ed. N.E. Mastorakis  
  **Place:** Massachusetts. EEUU

- **Title:** A Symbolic Portable Debugger for Compilers that generate C Code  
  **Authors:** J.M. Prieto, J.L. Arjona, R. Corchuelo, M. Toro, and D. Ruiz  
  **Reference:** Software and Hardware Engineering for the 21st Century. ISBN 960-8052-06-08  
  **Initial page:** 105  
  **Final page:** 110  
  **Date:** Ag-1999  
  **Editorial:** World Scientific Engineering Society. Ed. N.E. Mastorakis  
  **Place:** Massachusetts. EEUU

- **Title:** Extracting Information from the Web in a Multi-Agent Platform  
  **Authors:** J.L. Arjona, and R. Corchuelo  
  **Initial page:** 125  
  **Final page:** 136  
  **Date:** Jan-2001  
  **Editorial:** Catedral
• **Title:** An Approach Towards and Automatic Processing of Architectural Alternatives  
  **Authors:** O. Martín, A. Ruiz, D. Ruiz, J. Peña, and J.L. Arjona  
  **Initial page:** 13  
  **Final page:** 26  
  **Date:** Oct-2002  
  **Editorial:** Catedra  
  **Place:** Salamanca. España

• **Title:** Automatic Extraction of Semantically-Meaningful information from the Web  
  **Authors:** J.L. Arjona, A. Ruiz, J. Peña, D. Ruiz, and O. Martín  
  **Initial page:** 75  
  **Final page:** 87  
  **Date:** Oct-2002  
  **Place:** Salamanca. Spain
B.3.3 Conferences

- **Title:** Un depurador simbólico de código  
  **Congress:** IV Jornadas de Informática. Escuela de Verano  
  **Place:** Las Palmas. España  
  **Date:** Jul-1998  
  **Authors:** Rafael Corchuelo, José M. Prieto, José L. Arjona, Ignacio Rubio, Luisa C. Paredes, Amelia B. Dominguez, M. Victoria Rus, Jose A. Perez Castellanos, and Amador Durán

- **Title:** Selección Justa, Optimización y Superposición en el Contexto de TESORO  
  **Congress:** III Jornadas de Trabajo Menhir  
  **Publication:** Proceedings. Pages 75-86  
  **Place:** Murcia. España  
  **Date:** Nov-1998  
  **Authors:** R. Corchuelo, O. Martín, D. Ruiz, M. Toro, J.M. Prieto, and J.L Arjona

- **Title:** A Distributed Solution to Synchronous Multiparty Interaction  
  **Congress:** III IMACS/IEEE CSCC’99  
  **Publication:** Proceedings. Pages: 3511-3516  
  **Place:** Atenas. Grecia  
  **Date:** Jan-1999  
  **Authors:** R. Corchuelo, D. Ruiz, M. Toro, J.L. Arjona, and J.M. Prieto

- **Title:** A Symbolic Portable Debugger for Compilers that Generate C Code  
  **Congress:** III IMACS/IEEE CSCC’99  
  **Publication:** Proceedings. Pages: 3501-3506  
  **Place:** Atenas (Grecia)  
  **Date:** Jan-1999  
  **Authors:** J.M. Prieto, J.L. Arjona, R. Corchuelo, M. Toro, and D. Ruiz
• **Title:** Implementación de Interacciones entre Múltiples Participantes sobre una Máquina Virtual en Red  
**Congress:** Simposio Español sobre Informática Distribuida SEID’99  
**Place:** Santiago de Compostela. España  
**Date:** Feb-1999  
**Authors:** D. Ruiz, R. Corchuelo, M. Toro, A. Durán, and J.L. Arjona

• **Title:** Prototyping IPC on a Network Computer  
**Congress:** IV Jornadas de trabajo MENHIR  
**Publication:** Proceedings. Pages: 41-45  
**Place:** Burgos. España  
**Date:** May-1999  
**Authors:** R. Corchuelo, D. Ruiz, M. Toro, J.L. Arjona, and J.M. Prieto

• **Title:** Extracción de información en una plataforma multiagente  
**Congress:** Taller de trabajo sobre Métodos y Herramientas Desarrollo Aplicaciones  
**Publication:** Proceedings. Pages: 83-98  
**Place:** Almagro (Ciudad Real). España  
**Date:** Nov-2001  
**Authors:** J.L. Arjona, A. Ruiz-Cortés, and M. Toro

• **Title:** Orientando a Aspectos la captura de datos de Internet para activar Reglas de Decisión en Agentes Web  
**Congress:** VI Jornadas de Ingeniería del Software y Bases de Datos. JISBD 2001  
**Publication:** JISBD 2001 proceedings  
**Place:** Almagro (Ciudad Real). España  
**Date:** Nov-2001  
**Authors:** Rafael Corchuelo, Jesús S. Aguilar, J.L. Arjona, Miguel Toro, and José Riquelme
• **Title:** Automatic Extraction of Information from the Web  
  **Congress:** V Workshop Iberoamericano de Ingeniería de Requisitos y Desarrollo de Ambientes. IDEAS’02  
  **Place:** La Habana. Cuba  
  **Date:** Nov-2001  
  **Authors:** J.L. Arjona, R. Corchuelo, and M. Toro

• **Title:** A Practical Agent-Based Method to Extract Semantic Information from the Web  
  **Congress:** 14th International Conference on Advanced Information Systems Engineering (CAISE’02).  
  **Publication:** Springer Verlag, LNCS 2348. Pages: 697-700  
  **Place:** Toronto. Canada  
  **Date:** May-2002  
  **Authors:** J.L. Arjona, R. Corchuelo, A. Ruiz-Cortés, and M. Toro

• **Title:** Extraction of Semantically-Meaningful Information from the Web  
  **Congress:** 2nd International Conference on Adaptive Hypermedia and Adaptive Web Based Systems (AH’02)  
  **Publication:** Springer Verlag, LNCS 2347. Pages: 24-35  
  **Place:** Málaga. Spain  
  **Date:** May-2002  
  **Authors:** J.L. Arjona, R. Corchuelo, and A. Ruiz-Cortés

• **Title:** Towards Interaction Protocol Operations for Large Multi-Agent Systems  
  **Congress:** Second International Workshop on Formal Approaches to Agent-Based Systems  
  **Publication:** Springer-Verlag, LNAI.  
  **Place:** NASA-Goddard Space Flight Center, Greenbelt, MD, USA  
  **Date:** Nov-2002  
  **Authors:** J. Peña, R. Corchuelo, and J. L. Arjona
• Title: A Top Down Approach for MAS Protocol Descriptions
  Congress: ACM Symposium on Applied Computing SAC’03
  Publication: ACM Press. Pending to publish.
  Place: Melbourne, Florida, USA
  Date: March-2003
  Authors: J. Peña, R. Corchuelo, and J. L. Arjona

• Title: Coping with Web Knowledge
  Congress: First International Atlantic Web Intelligence Conference. AWIC 2003
  Publication: Springer Verlag, LNAI 2663. p. 165 ff.
  Place: Madrid, Spain
  Date: May-2003
  Authors: J.L. Arjona, R. Corchuelo, J. Peña, and D. Ruiz

• Title: Knowledge Channels. Bringing the Knowledge on the Web to Software Agents
  Publication: Pending to publish.
  Place: Velden. Austria
  Date: May-2003
  Authors: J.L. Arjona, R. Corchuelo, D. Ruiz, J. Peña and M. Toro
B.4 Official Research Projects

1. **WebPlus**
   - **Full name:** Web Plus, New Web Services Generation
   - **Code:** FIT-150100-2001-78
   - **Support:** Spanish Ministry of Science and Technology
   - **Budget:** 456 770 €
   - **Dates:** 2001 - 2003
   - **Coordinators:** Jesús Bermejo and Antonio Ruiz
   - **Description:** The aim of the project is to study web services technologies, particularly focusing on Quality aspects. Current web services repositories and their capability to add quality parameters are being studied. We are defining a quality model to build web services-based systems using the UDDI repository (Universal Description, Discovery and Integration of business for the web).

2. **GEOZOCO**
   - **Full name:** New Methods and Tools for Automating the Development of Electronic Commerce Applications and its Integration with Geographic Information Systems
   - **Code:** TIC2000-1106-C02-01 (18.13.03.30.02 2001/72)
   - **Support:** Spanish Interministerial Commission of Science and Technology
   - **Budget:** 70 000 €
   - **Dates:** 2000 - 2003
   - **Coordinator:** Dr. Miguel Toro
   - **Description:** One of the main aims of the software industry consists of producing high-quality applications at costs that should be as reduced as possible. Unfortunately, building such applications keeps on being quite a handmade activity that lacks standardised, consolidated enough automated production methods.

Furthermore, the Internet has entailed a great revolution in this industry, and it is driving electronic commerce activities at an ever-increasing pace. This has implied adapting existing methods and tools to the new resources and means the Internet provides, but at such a pace that they do not have enough time to consolidate before new proposals sprout out. The Internet has also pulled down geographic fences, and many enterprises have benefited from this fact because they have been able to increase their market share at quite reasonable costs, which has also
brought geographic information systems out (GIS). The main feature of a GIS is its ability to deal with information relating geographic entities such as countries, regions, cities, and so on. This feature transforms them into useful tools in the field of electronic commerce because, on the one hand, they allow users to have access to commercial offers, and, on the other hand, they are essential to locate and tell the customers where the nearest storehouse or local office is, to design delivery routes dynamically, to estimate service times, and so on.

In this research project, our goal is to develop a methodological framework for the construction of electronic-commerce applications able to adapt to the continuous changes in this field. This framework must be supported by a set of tools for developing high-quality software able to deal with problems at an adequate abstraction level. The goal is to begin with an architectural description obtained from the requirements analysis that should be refined by means of standard, firm industrial techniques: XML, aspect-orientation, components, middlewares, and so on. Certainly, since it seems that the Internet is going to embrace every aspect of enterprises, this framework cannot pass over GISs, which are going to be used to deal with geographic information relating commercial activities so that the enterprises may benefit from it.

3. **WEST**

It’s a project funded by the Iberoamerican Scientific and Technological Programme for the Development. We work with a number of academic partners on web-oriented software technology. We’re involved in Task 4, which focuses on component-oriented web software construction.
B.5 I+D Organisation Activities

1. Workshop on Object-Oriented Business Solutions (WOOBS’01)
   Organisation Committee Member
   International Event
   Jun-2001

2. Reunión de trabajo sobre métodos y herramientas para el desarrollo de aplicaciones de comercio electrónico (ZOCO)
   Organisation Committee Member
   National event
   Nov-2001

3. Workshop on Object-Oriented Business Solutions (WOOBS’02)
   Organisation Committee Member
   International Event
   Jun-2002

4. Simposio de Informática y Telecomunicaciones (SI+T 02)
   Organisation Committee Member
   Nacional
   Sept-2002

5. ZOCO: Métodos y Herramientas para el Comercio Electrónico
   Program Committee Member
   National Event
   Nov-2002

6. Taller en Sistemas Hipermedia Colaborativos y Adaptativos
   Program Committee Member
   National Event
   Nov-2002
B.6 I+D with enterprises


Company: Isotrol, S.A.
Duration: 20 February 2002 – nowdays
Head of research: José Luis Arjona y Antonio José Sáenz
Number of researches: 4

Description: This project aims at the definition of a software development methodology for the instantiation of agile process in the context of the international norm ISO 9001:2000.

The project was expected to fulfill the following goals:

- To accept the need of clients to evolve requirements during the project lifecycle.
- To ensure the production of high-quality software.
- To increase productivity.
- To guarantee that all software developments follows a predictable model. That is, it does not depend on a project manager’s skills.

A software development methodology, that builds on the Enterprise Unified Process (EUP), was defined. It has been tailored with the principles and practices of Agile Modeling (AM), improving productivity in modeling disciplines. It has also been tailored with the principles and practices of Extreme Programming (XP), improving productivity in the implementation discipline. The developed methodology has been validated with several software projects, and Isotrol, Inc become the first European company to achieve the ISO 9001 certification of quality using of agile practices.
Bibliography


