UNIVERSITÀ DEGLI STUDI DI NAPOLI FEDERICO II

BIOINVIEW: A SYSTEM FOR INTEGRATING HETEROGENEOUS AND DISTRIBUTED BIOLOGICAL DATA SOURCES

PASQUALE CAPASSO

DOTTORATO DI RICERCA IN INGEGNERIA INFORMATICA ED AUTOMATICA

XX CICLO

Tutore:
Prof. Antonio Picariello

Coordinatore del Dottorato:
Prof. Luigi P. Cordella

DIPARTIMENTO DI INFORMATICA E SISTEMISTICA
NAPOLI, NOVEMBER 2007

To my Mother
and my Father..
in my heart forever

To my Brother..
best friend and playmate

To all the Friends and
Relatives (Cinzia Tornatore included)..
for loving me as much as i love them (at least)

To The One..

Contents

List of Figures

To Fabio..

I only had few months to know him,

Nevertheless he taught me how to deal with

Every challenge in life with a smile on the face..

Thanks, my friend.. I'll never forget you!

"Science is like Sex: sometimes something useful
Comes out, but that's not the reason we are doing it."

Richard Feynman.

Introduction

The aim of the system presented in this thesis is to provide life scientists with an efficient and effective tool for browsing, querying and integrating distributed biological data sources, which could be transparent to the format heterogeneity and the distribution of the data.

To this purpose, we had to face several problems:

1. *how to locate and access the required data on the distributed sources;*
2. *how to resolve format heterogeneity;*
3. *how to store and translate intermediate results;*
4. *how to automate the generation of the mappings of the sources onto the global model.*

The importance of such a heavy work is clearly motivated by the pressing need for integration deriving from the exponential growth of data that has characterized the biological research domain in the last two decades, caused in turn by the application of new sequencing and analytical techniques that since the early 90s have been providing biologists with many new complete genomes every year and a great deal of information concerning the interaction between proteins in physiological processes.

Being Biology a knowledge-based discipline, where prediction and analysis are based on comparing new data to the existing knowledge, the need for systems and tools able to help biologists in filtering and integrating this big flood of data soon became urgent. This led to the birth of a significant number of public databases[1]*, certainly helping biologists in managing such a plethora of information, but very often posing new challenges, in terms of integration and information sharing. These sources have indeed complete autonomy, continually extending their coverage, and are poorly integrated and difficult (if not only time consuming) to use together.*

In figures 1, 2, we show a sketch of the entries of two different protein sequence databases (UniProtKB/Swiss-Prot and PIR-PSD, respectively), corresponding to the same protein (14 kDa proline-rich protein DC2.15 precursor). You could easily see how the format and the schema could be different, even for such strongly related sources (the entries shown in the pictures even reference one each other).

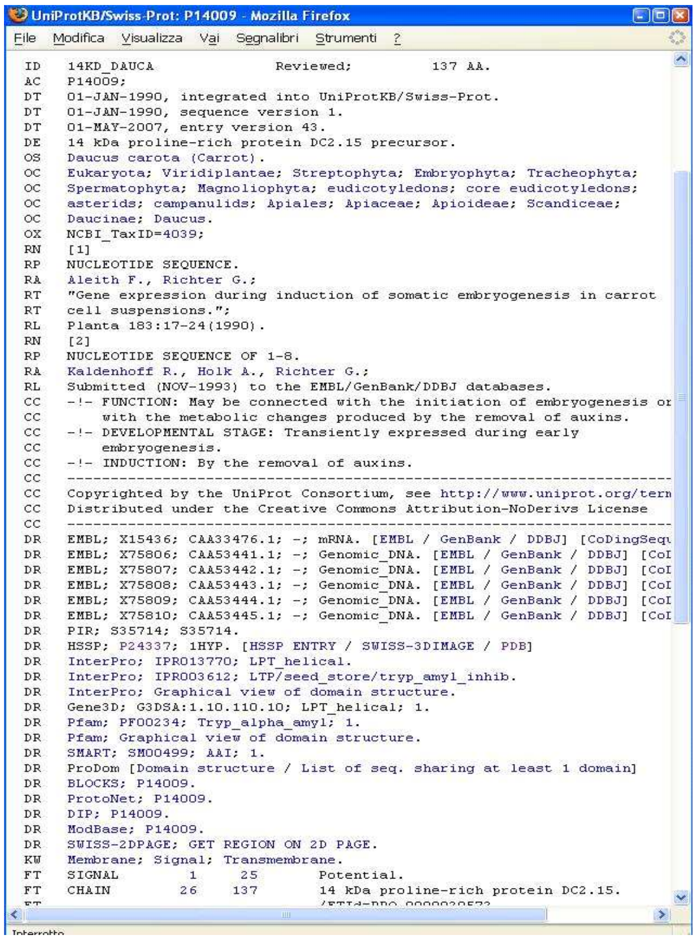
This heterogeneity of the data sources greatly complicates the retrieval tasks biologists need to perform [46]; to accomplish these tasks, usually a biologist needs to:

1. *Construct his/her own view of the meta-data in each source and the instances covered by that source, resolving any semantic heterogeneities between the sources;*
2. *Construct the various parts of the request in the different formats and terms required by the different sources;*
3. *Locate and communicate with the sources, and process intermediate results into appropriate input formats for successive stages;*

[1] *Usually, these public sources are not really databases in the conventional sense of the term, in that they do not have a separate schema containing meta-data (or if they do, it is not always publicly accessible). Some of them are actually no more than tools, processes, or Internet flat files containing embedded meta-data, with a limited set of services accessible through suitable interfaces*

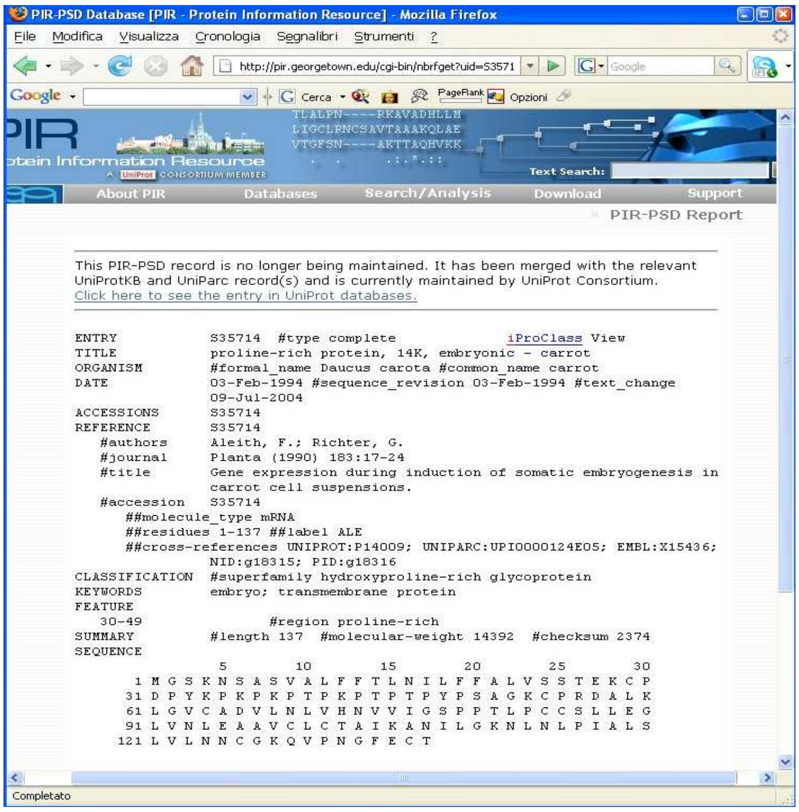
4. *Inter-operate between resources, planning a suitable series of requests that get from each resource the relevant information;*

5. *Optimize the query process (the user has to choose among different execution plans, with different efficiency).*


UniProtKB/Swiss-Prot: P14009 - Mozilla Firefox
File Modifica Visualizza Vai Segnalibri Strumenti ?
Interrotto

```
ID   14KD_DAUCA              Reviewed;         137 AA.
AC   P14009;
DT   01-JAN-1990, integrated into UniProtKB/Swiss-Prot.
DT   01-JAN-1990, sequence version 1.
DT   01-MAY-2007, entry version 43.
DE   14 kDa proline-rich protein DC2.15 precursor.
OS   Daucus carota (Carrot).
OC   Eukaryota; Viridiplantae; Streptophyta; Embryophyta; Tracheophyta;
OC   Spermatophyta; Magnoliophyta; eudicotyledons; core eudicotyledons;
OC   asterids; campanulids; Apiales; Apiaceae; Apioideae; Scandiceae;
OC   Daucinae; Daucus.
OX   NCBI_TaxID=4039;
RN   [1]
RP   NUCLEOTIDE SEQUENCE.
RA   Aleith F., Richter G.;
RT   "Gene expression during induction of somatic embryogenesis in carrot
RT   cell suspensions.";
RL   Planta 183:17-24(1990).
RN   [2]
RP   NUCLEOTIDE SEQUENCE OF 1-8.
RA   Kaldenhoff R., Holk A., Richter G.;
RL   Submitted (NOV-1993) to the EMBL/GenBank/DDBJ databases.
CC   -!- FUNCTION: May be connected with the initiation of embryogenesis or
CC       with the metabolic changes produced by the removal of auxins.
CC   -!- DEVELOPMENTAL STAGE: Transiently expressed during early
CC       embryogenesis.
CC   -!- INDUCTION: By the removal of auxins.
CC   -----------------------------------------------------------------------
CC   Copyrighted by the UniProt Consortium, see http://www.uniprot.org/tern
CC   Distributed under the Creative Commons Attribution-NoDerivs License
CC   -----------------------------------------------------------------------
DR   EMBL; X15436; CAA33476.1; -; mRNA. [EMBL / GenBank / DDBJ] [CoDingSequ
DR   EMBL; X75806; CAA53441.1; -; Genomic_DNA. [EMBL / GenBank / DDBJ] [CoI
DR   EMBL; X75807; CAA53442.1; -; Genomic_DNA. [EMBL / GenBank / DDBJ] [CoI
DR   EMBL; X75808; CAA53443.1; -; Genomic_DNA. [EMBL / GenBank / DDBJ] [CoI
DR   EMBL; X75809; CAA53444.1; -; Genomic_DNA. [EMBL / GenBank / DDBJ] [CoI
DR   EMBL; X75810; CAA53445.1; -; Genomic_DNA. [EMBL / GenBank / DDBJ] [CoI
DR   PIR; S35714; S35714.
DR   HSSP; P24337; 1HYP. [HSSP ENTRY / SWISS-3DIMAGE / PDB]
DR   InterPro; IPR013770; LPT_helical.
DR   InterPro; IPR003612; LTP/seed_store/tryp_amyl_inhib.
DR   InterPro; Graphical view of domain structure.
DR   Gene3D; G3DSA:1.10.110.10; LPT_helical; 1.
DR   Pfam; PF00234; Tryp_alpha_amyl; 1.
DR   Pfam; Graphical view of domain structure.
DR   SMART; SM00499; AAI; 1.
DR   ProDom [Domain structure / List of seq. sharing at least 1 domain]
DR   BLOCKS; P14009.
DR   ProtoNet; P14009.
DR   DIP; P14009.
DR   ModBase; P14009.
DR   SWISS-2DPAGE; GET REGION ON 2D PAGE.
KW   Membrane; Signal; Transmembrane.
FT   SIGNAL        1     25       Potential.
FT   CHAIN        26    137       14 kDa proline-rich protein DC2.15.
```

Figure 1: Example of UniProtKB/Swiss-Prot entry.


PIR-PSD Database [PIR - Protein Information Resource] - Mozilla Firefox
File Modifica Visualizza Cronologia Segnalibri Strumenti ?
http://pir.georgetown.edu/cgi-bin/nbrfget?uid=S35714
Protein Information Resource
A UniProt CONSORTIUM MEMBER
Text Search:
About PIR
Databases
Search/Analysis
Download
Support
PIR-PSD Report
Completato

This PIR-PSD record is no longer being maintained. It has been merged with the relevant UniProtKB and UniParc record(s) and is currently maintained by UniProt Consortium.
Click here to see the entry in UniProt databases.

```
ENTRY           S35714  #type complete             iProClass View
TITLE           proline-rich protein, 14K, embryonic - carrot
ORGANISM        #formal_name Daucus carota #common_name carrot
DATE            03-Feb-1994 #sequence_revision 03-Feb-1994 #text_change
                09-Jul-2004
ACCESSIONS      S35714
REFERENCE       S35714
   #authors     Aleith, F.; Richter, G.
   #journal     Planta (1990) 183:17-24
   #title       Gene expression during induction of somatic embryogenesis in
                carrot cell suspensions.
   #accession   S35714
      ##molecule_type mRNA
      ##residues 1-137 ##label ALE
      ##cross-references UNIPROT:P14009; UNIPARC:UPI000124E05; EMBL:X15436;
                NID:g18315; PID:g18316
CLASSIFICATION  #superfamily hydroxyproline-rich glycoprotein
KEYWORDS        embryo; transmembrane protein
FEATURE
   30-49               #region proline-rich
SUMMARY         #length 137  #molecular-weight 14392  #checksum 2374
SEQUENCE
                 5         10        15        20        25        30
      1 M G S K N S A S V A L F F T L N I L F F A L V S S T E K C P
     31 D P Y K P K P K P T P K P T P T P Y P S A G K C P R D A L K
     61 L G V C A D V L N L V H N V V I G S P P T L P C C S L L E G
     91 L V N L E A A V C L C T A I K A N I L G K N L N L P I A L S
    121 L V L N N C G K Q V P N G F E C T
```

Figure 2: Example of PIR-PSD entry.

To automate these steps and make the heterogeneity of the data as transparent as possible to the user, we relied on an ontology-based integration approach. The different sources are suitably wrapped, and a mediator provides for linking mechanisms between these sources, in other words it is responsible of maintaining the mappings of the sources onto the model ontology; of processing, dividing, and planning the execution of the queries presented by the user in a source-independent language; of elaborating intermediate results and presenting the retrieved entries to the user.

As Hernandez [80] underlines, the 'ideal system' *should automate a maximum number of tasks, at the same time reducing the number of interactions users need to perform to find what they are looking for, and providing enough flexibility to the user as well as displaying the provenance of data. In particular, Hernandez points out the great importance of automating the process of describing the sources (something that in existing systems is mostly obtained by manual analysis of domain experts) to reduce the cost and time necessary to develop full-scale systems that can keep up with the pace at which biological data are generated. This latter aspect is what we believed to be deserved greater attention, being at the same time very challenging and fundamental for a well functioning integration system.*

On the other hand, since the beginning of the design phase it was clear that the automation of the description of the sources had not to prevail on the correctness and the completeness of the representation. This observation soon led to the conclusion that a completely automatic mapping process was in practice unfeasible, being many sources' schema characterized by attributes (records) with very complex (often not even human-understandable) names.

Another point that we considered critical was the flexibility that we should have given to the system, namely to which extent the user should have been left free to set the parameters of the system (sources to be queried, format of the results, or even the query language, the minimum/maximum number of retrieved entries, . . .) and in general to interact with the system itself.

The thesis is organized in five chapters. In chapter one an overview of the literature on the use of ontologies and data integration techniques in the field of molecular *biology is presented; in chapter two we present modeling issues and the choices we made, with respect to every part of the system; then in the third chapter implementation aspects are discussed; and finally, in chapter four and five, experimental results are reported and discussed, followed by our conclusions and observations about future works.*

An important note: the ideas behind most of the work presented in this thesis arose from the analysis of the literature and the consequent discussions I had with my advisor, Antonio Picariello, and with Antonio Penta, a colleague and friend who deserves my thanks here for his continuous support and the capability of being interested in pretty much everything

"What's in a name? That which we call a rose,
By any other name would smell as sweet."

William Shakespeare.

Chapter 1

Related Works

In this chapter, some theoretical aspects of knowledge representation and integration are presented, with particular attention to the use of ontologies.

Furthermore, we are going to present the main results of the research in the field of Data Integration (with particular focus on Ontology-based algorithms and tools), and the application of these results to solving the integration challenges researchers have to address every day in the molecular biology field.

1.1 Knowledge Representation

The goal of knowledge representation is to create schemes that allow information to be efficiently stored, modified, and reasoned with. In this section, we will consider a number of representations and formalisms (i.e., knowledge representation languages) that are particularly relevant to a distributed and dynamic environment.

Although knowledge representation is one of the central and, in some ways, most familiar concepts in computer science (more precisely in the field of AI), the most fundamental question about it - What is it? - has rarely been answered directly.

A good answer to this question, that we could call a *'functional'* answer, is provided in [10], where Davis et al. define a knowledge representation on the basis of five different roles the representation plays.

According to Davis, a knowledge representation is first and foremost a surrogate, a substitute for the thing itself, that is used to enable an entity to determine consequences by thinking rather than acting, that is by reasoning about the world rather than taking action in it.

Second, it is a set of ontological commitments, i.e. an answer to the question *'In what terms should I think about the world?'*. Third, it is a fragmentary theory of intelligent reasoning expressed in terms of three components: (1) the representation's fundamental conception of intelligent reasoning, (2) the set of inferences that the representation sanctions, and (3) the set of inferences that it recommends.

Fourth, it is a medium for efficient computation, that is, the computational environment in which thinking is accomplished. One contribution to this pragmatic efficiency is supplied by the guidance that a representation provides for organizing information to help making the recommended inferences. Fifth, it is a medium of human expression, i.e. a language in which we say things about the world.

Note that each role requires something slightly different from a representation; each accordingly leads to an interesting and different set of properties that we want a representation to have.

For example, the first property of any representation to be a surrogate of the represented information implies the need for some specification of its intended meaning (*'semantics'* for the representation). At the same time, different degree of fidelity of the representation (i.e., how close to its recipient in the real world is the surrogate representing it) are possible, each representation inevitably carrying with its status of surrogate a not well-definable error.

For an exhaustive treatment of the properties a knowledge representation may have, see [10].

We proceed now to present some formalisms and representations that are particularly relevant [51] (see [26] for further reading) .

Semantic Networks

One of the oldest knowledge representation formalisms is semantic networks [1]. In a semantic net, each concept is represented by a node in a graph. Concepts that are semantically related are connected by arcs, which may or may not be labeled. In such a representation, meaning is implied by the way a concept is connected to other concepts.

It is rather common to use two arcs for representing abstractions. An *is-a* arc indicates that one concept is subclass of another, while an *instance-of* arc indicates that a concept is an example of another concept. These arcs have correlations in basic set theory: *is-a* is like the *subset* relation and *instance-of* is like the *element of* relation. The collection of *is-a* arcs specifies a partial order on classes; this order is often called a taxonomy or categorization hierarchy. The taxonomy can be used to generalize a concept to a more abstract class or to specialize a class to its more specific concepts.

As demonstrated by the popularity of Yahoo and the Open Directory, taxonomies are clearly useful for aiding a user in locating relevant information on the Web. However, these directory taxonomies often deviate from the strict subset semantics followed by modern knowledge representation systems, making them less useful for automated reasoning.

Frame Systems

In the 1970's, Minsky [2] introduced frame systems. In the terminology of such systems, a frame is a named data object that has a set of slots, where each slot represents a property or attribute of the object. Slots can have one or more values (called fillers), some of which may be pointers to other frames.

Since each frame has a set of slots that represent its properties, frame systems are usually considered to be more structured than semantic networks. However, it has been shown that frame systems are isomorphic to semantic networks.

Notable examples of frame-based knowledge representation languages are KRL [3], and KL-ONE [5].

Description Logics

Description logics focus on the definitions of terms in order to provide more precise semantics than semantic networks or earlier frame systems. Term definitions are formed by combining concepts and roles that can provide either necessary and sufficient conditions or just necessary conditions.

An important feature of description logic systems is the ability to perform automatic classification, that is, automatically insertion of a given concept at the appropriate place in the taxonomy.

The advantages of descriptions logics are they have well-founded semantics and the factors that affect their computational complexity are well understood, but it is unclear whether their inferential capabilities are the right ones for that huge distributed environment called Web.

First Order Logics

First-order logic (FOL), also known as predicate calculus or predicate logic, is a well-understood formalism for reasoning. Although the logic and knowledge representation communities are distinct, the expressivity of FOL nevertheless makes it a powerful knowledge representation language.

From the perspective of FOL, the world consists of objects and the relations that hold between them. A FOL language consists of logical and non-logical symbols. The logical symbols represent quantification, implication, conjunction and disjunction; while the non-logical symbols are constants, predicates, functions, and variables. Constant, variable and function symbols are used to build terms, which can be combined with predicates to construct formulas.

The semantics of FOL are given by Tarski's model theory, where the concepts of 'Interpretation' and satisfaction of formulas by a given interpretation are provided.

An introductive and a more detailed treatment of FOL can be found in [6], and [7], respectively.

Ontology

In order for information from different sources to be integrated, there needs to be a shared understanding of the relevant domain. Knowledge representation formalisms provide structures for organizing this knowledge, but provide no mechanisms for sharing it. Ontologies provide a common vocabulary to support the sharing and reuse of knowledge.

More on ontologies and their use in the fields of semantic web and data integration will be given in some of the next sections.

Context Logic

One of the problems with knowledge representation is that when we try to conceptualize some part of the world, we must make some simplifying assumptions about its structure. If we then try to combine knowledge bases (or logical theories), differences in their implicit, underlying assumptions may have unintended side-effects. Context logic (see [9], [13], [8]) proposes to solve this problem by explicitly placing each assertion in a context, where the context includes the assumptions necessary for the assertion to be true.

In context logic, contexts are first-class objects that can be used in propositions. Propositions of the form $ist(c, p)$ are used to indicate that proposition p is true in context c. A particular individual i can be excluded from the scope of a context c by stating $\neg presentIn(c, i)$.

The reification of context also makes it possible to combine information from many contexts. For example, one may wish to reuse parts of one context in another or make statements that are simultaneously true in a set of contexts. Statements that achieve these effects are called lifting axioms.

Another issue raised by context logic is that different contexts may contain mutually inconsistent assertions. Such situations should not lead to inconsistency of the entire knowledge base. Instead, context logic only requires a context to be locally consistent. This issue is of direct relevance to the Semantic Web, where knowledge is being provided by many users who may have inconsistent assumptions.

Note that Ontologies and context logic are closely related. Each context is an ontology, and ontology inclusion could be one particular type of lifting axiom.

1.2 Semantic Web and Ontology

The Semantic Web is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, but also in a format that can be read and used by software agents, with the very purpose of permitting these agents to find, share and integrate information more easily [59].

On the Semantic Web, computers do the browsing (and searching, and querying, and much more) for us. The Semantic Web enables computers to seek out the knowledge distributed throughout the Web, mesh it, and then take action based on it. Take an analogy: the current web is a decentralized platform for distributed *presentations*, while the Semantic Web is a decentralized platform for distributed *knowledge* [92].

There, of course, is knowledge on the current web, but it is off limits to computers. Consider a Wikipedia page, which might convey many information to the human reader, but the computer displaying the page only sees presentation markup. To the extent that computers make sense of HTML, images, Flash, etc., it is almost always for the simple purpose of creating a presentation for the end user. The real content, the knowledge the files are conveying to the human, is opaque to the computer.

What is meant by *'semantic'* in Semantic Web is not that computers are going to understand the meaning of anything, but that the logical pieces of meaning can be mechanically manipulated by a machine to useful *human* ends.

At its core, the semantic web comprises a philosophy, a set of design principles, collaborative working groups, and a variety of enabling technologies. Some elements of the semantic web are expressed as prospective future possibilities that have yet to be implemented or realized. Other elements of the semantic web are expressed in formal specifications [60].

Some of these include Resource Description Framework (RDF), a variety of data interchange formats (e.g. RDF/XML, N3, Turtle, N-Triples), and notations such as RDF Schema (RDFS) and the Web Ontology Language (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

The basis for the augmented functionality of the Semantic Web are:

- a global naming scheme (URIs);
- a standard syntax for describing data (RDF);
- a standard means of describing the properties of that data (rdf-schema);
- a standard means of describing relationships between data items (ontology);
- the means to support trust and security.

1.2.1 Global naming scheme

If any Semantic Web application is to be able to access and use data from any other such application, every data object and every data schema/model must have a unique and universal means of identification. These identifiers are called URIs (Universal Resource Identifiers).

1.2.2 Standard Syntax - RDF

The computer industry has agreed to use XML (Extensible Markup Language) to represent not only human readable documents, but data in general. The XML standards give a syntactic structure for describing data.

Unfortunately, XML can be used in many different ways to describe the same data. This makes it too open and arbitrary to support the type of widespread and ad hoc data integration envisaged for the Semantic Web. The Semantic Web vision proposes to represent machine processable information using RDF (Resource Description Framework), which extends XML. RDF defines a general common data model that adheres to web principles (see appendix B). The W3C are strong supporters of this approach.

RDF was originally created in 1999 as a standard on top of XML for encoding meta-data (literally, *data about data*). meta-data is, of course, things like 'who authored a web page', or 'what date a blog entry was published', information that is in some sense secondary to the content already on the regular web.

Since then, and perhaps especially after the updated RDF spec in 2004, the scope of RDF has really evolved into something greater. The most exciting uses of RDF are not in encoding information about web resources, but *information about* and *relations between* things in the real world: people, places, concepts, etc.

From the web programmer's point of view, RDF provides a consistent, standardized way of describing and querying internet resources, from text pages and graphics to audio files and video clips. It gives syntactic interoperability, and provides the base on top of which building the Semantic Web.

As for what RDF can do, it basically defines a directed graph of relationships. These are represented by object-attribute-value triples, i.e. an object O has an attribute A with value V, often written as $A(O, V)$. For instance, *telnet(janet_bruten, 3128700* represents the fact that the person object *Janet Bruten* has the *telnet* number *312-8700*.

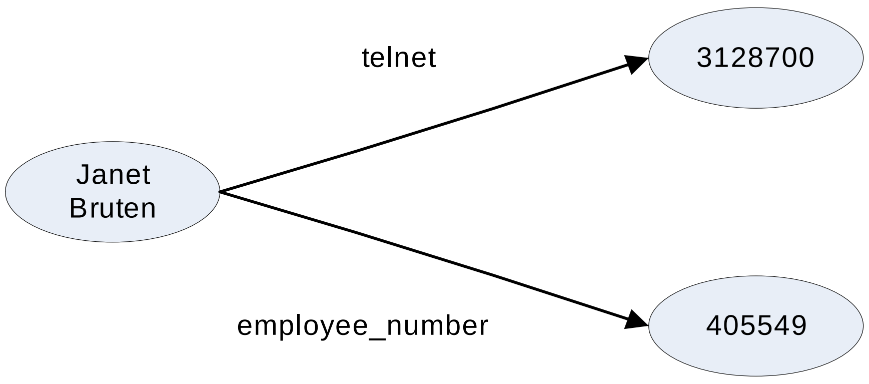


Figure 1.1: An example RDF triple; the sketched triple means '*Janet Bruten* has the *telnet* number *312-8700* and *employee* number *405/549*'.

1.2.3 Describing properties - RDF Schema

RDF itself is a composable and extensible standard for building data models. To support the definition of a specific vocabulary for a data model, which can itself be published, another layer is required. RDF schema allows a designer to define and publish the vocabulary used by an RDF data model, i.e define the data objects and their attributes. For instance, it might define that people have a phone attribute. RDF Schema (or RDFS) also uses class and subclass, so that *hp_employee* could be defined as a sub-class of *person*.

Both RDF and RDFS are based on XML and XML-Schema. The existence of standards for describing data (RDF) and data attributes (RDFS) enables the development of a set of readily available tools to read and exploit data from multiple sources. The degree to which different applications can share and exploit data is sometimes termed *syntactic interoperability*.

The more standardised and widespread these data manipulation tools are the higher the degree of syntactic interoperability, and the easier and more attractive it becomes to use the Semantic Web approach as opposed to a point solution.

1.2.4 Describing relationships between data items - Ontology

If data is to be truly *'understandable'* by multiple applications, and therefore become information, semantic interoperability is required. Syntactic interoperability is all about parsing data correctly. Semantic interoperability requires mapping between terms, which in turn requires content analysis. This requires formal and explicit specifications of domain models, which define the terms used and their relationships. Such formal domain models are sometimes called *ontologies*.

As discussed by Guarino and Giaretta [18], the meaning of the term ontology is often vague. It was first used to describe the philosophical study of the nature and organization of reality. In AI, the most cited definition is due to Tom Gruber [11], [12]:

"An ontology is an explicit specification of a conceptualization". In this definition, along the lines of Genesereth and Nilsson [6], the conceptualization is the couching of knowledge about the world in terms of entities (things, the relationships they hold and the constraints between them), the specification is the concrete representation of this conceptualization.

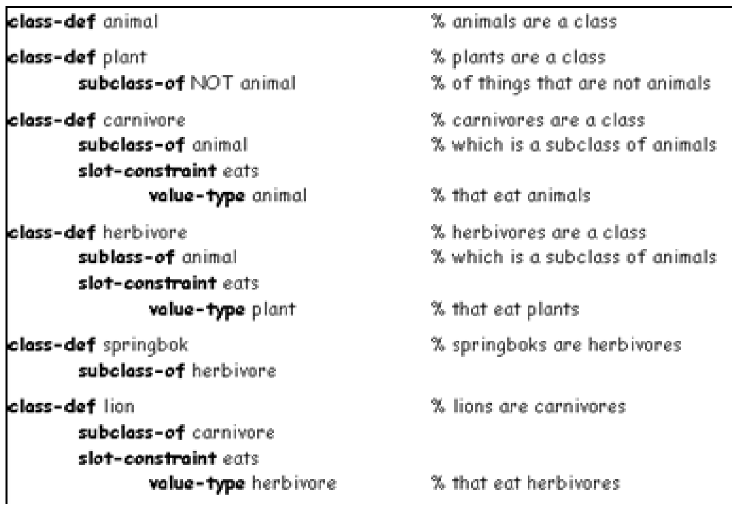
Guarino and Giaretta argue that Genesereth and Nilsson's definition of conceptualization should not be used in defining ontology, because it implies that a conceptualization represents a single state of affairs (i.e., it is an extensional structure), while an ontology should provide terms for representing all possible states of affairs with respect to a given domain.

In a later paper [27], Guarino provides the following definition for an ontology: *"An ontology is a logical theory accounting for the intended meaning of a formal vocabulary, i.e., its ontological commitment to a particular conceptualization of the world. The intended models of a logical language using such a vocabulary are constrained by its ontological commitment. An ontology indirectly reflects this commitment (and the underlying conceptualization) by approximating these intended models"*.

Ontologies generally define data models in terms of [95]:

- Individuals: the basic or *'ground level'* objects;
- Classes: sets, collections, or types of objects;
- Attributes: properties, features, characteristics, or parameters that objects can have and share;
- Relations: ways that objects can be related to one another;
- Events: the changing of attributes or relations;

For instance, we might define a herbivore to be a subclass of animals that eats plants. Figure 1.2 shows a very simple example ontology for animals.



```
class-def animal                        % animals are a class
class-def plant                         % plants are a class
        subclass-of NOT animal          % of things that are not animals
class-def carnivore                     % carnivores are a class
        subclass-of animal              % which is a subclass of animals
        slot-constraint eats
                value-type animal       % that eat animals
class-def herbivore                     % herbivores are a class
        sublass-of animal               % which is a subclass of animals
        slot-constraint eats
                value-type plant        % that eat plants
class-def springbok                     % springboks are herbivores
        subclass-of herbivore
class-def lion                          % lions are carnivores
        subclass-of carnivore
        slot-constraint eats
                value-type herbivore    % that eat herbivores
```

Figure 1.2: A simple example ontology for animals.

Over the years a vast amount of research has been carried on how to represent and reason about knowledge. In Europe funding has been heavily concentrated on the development of OIL (Ontology Inference Layer), a language for defining ontologies.

In the US, DARPA funded a somewhat similar project called DAML (Distributed Agent Markup Language). More recently these activities have been combined into a project to work on a merged ontology language, DAML+OIL.

In late 2001 the W3C set up a working group called WebOnt to define an ontology language for the Web, based on DAML+OIL. All of these ontology languages aim to provide developers with a way to formally define a shared conceptualization of a domain. They encompass both a means of representing the domain and a means of reasoning about that representation. In the case of DAML+OIL the latter is Description Logic.

Regardless of the language in which they are expressed, contemporary ontologies share many structural similarities. As mentioned above, most ontologies describe individuals (instances), classes (concepts), attributes (and relations).

Individuals: Individuals (instances) are the basic, *'ground level'* components of an ontology. The individuals in an ontology may include concrete objects such as people, animals, tables, automobiles, molecules, and planets, as well as abstract individuals such as numbers and words.

Strictly speaking, an ontology need not include any individuals, but one of the general purposes of an ontology is to provide a means of classifying individuals, even if those individuals are not explicitly part of the ontology.

Classes: Classes (Concepts) are abstract groups, sets, or collections of objects. They may contain individuals, other classes, or a combination of both. Some examples of classes are:

- *Person*, the class of all people;
- *Molecule*, the class of all molecules;
- *Number*, the class of all numbers;
- *Class*, representing the class of all classes;

Ontologies vary on whether classes can contain other classes, whether a class can belong to itself, whether there is a universal class (that is, a class containing everything), etc. Sometimes restrictions along these lines are made in order to avoid certain well-known logical paradoxes.

Attributes and Relations: Objects in the ontology can be described by assigning attributes to them. Each attribute has at least a name and a value, and is used to store information that is specific to the object it is attached to. For example the *'Ford Explorer'* object has attributes such as:

- Name: Ford Explorer;
- Number-of-doors: 4
- Engine: 4.0L, 4.6L
- Transmission: 6-speed

The value of an attribute can be a complex data type; in the example above, the value of the attribute called Engine is a list of values, not just a single value.

If you do not define attributes for the concepts you have either a taxonomy (if hyponym relationships exist between concepts) or a controlled vocabulary. These are useful, but are not properly true ontologies.

An important use of attributes is to describe the relationships (also known as relations) between objects in the ontology. A relation can be seen as an attribute whose value is another object in the ontology.

Though many existing ontologies make use of the sole 'is-a' and 'part-of' relations, any kind of relation can be represented, to further refine the semantics they model.

These relations are often domain-specific and are used to answer particular types of question.

Building an Ontology

Although there is some collective experience in developing and using ontologies, there exists no standardized methodologies for building them. The most well-known ontology construction guidelines were developed by Gruber [12] to encourage the development of more reusable ontologies. Some attempts to develop comprehensive ontology building methodology were lately made [20], [22], and a survey of these techniques can be found in [28].

In building an ontology, most distinguishes between an informal stage, where the ontology is sketched out using either natural language descriptions or some diagram technique, and a formal stage where the ontology is encoded in a formal knowledge representation language, which is machine computable.

The life cycle of the overall methodology is depicted in fig.1.3. The main stages of the process are [42]:

Identification of Purpose and Scope: A well-characterized requirements specification is important to the design, evaluation and reuse of an ontology.

Knowledge Acquisition: Sources span the complete range of knowledge holders: specialist biologists; database meta-data; standard textbooks; research papers;

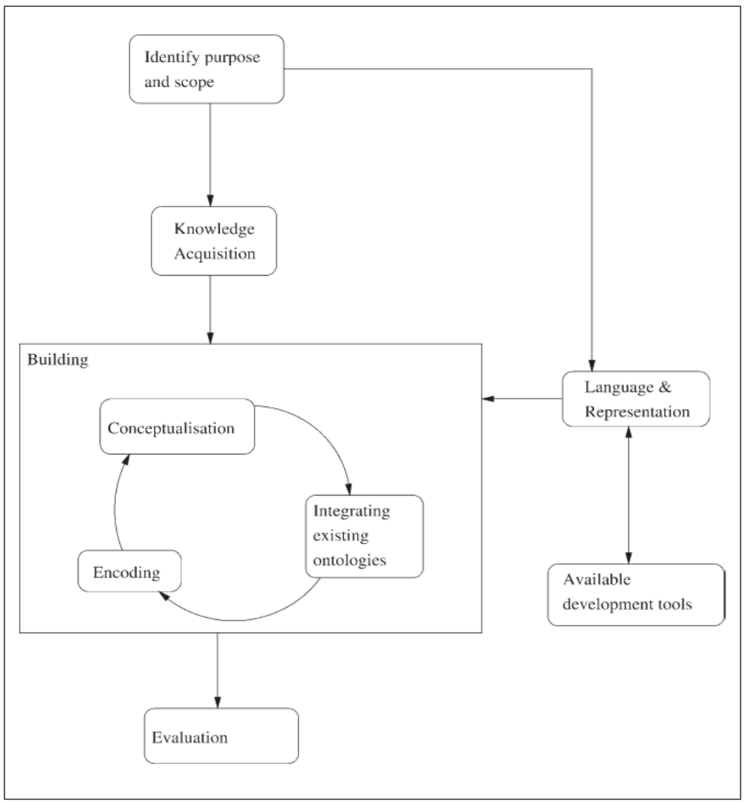

Identify purpose and scope
Knowledge Acquisition
Building
Conceptualisation
Integrating existing ontologies
Encoding
Language & Representation
Available development tools
Evaluation

Figure 1.3: Ontology Building Life Cycle

and other ontologies.

Conceptualization: identifying the key concepts that exist in the domain, their properties and the relationships between them; identifying natural language terms to refer to such concepts, relations and attributes; structuring domain knowledge into explicit conceptual models.

Integration of Existing Ontologies: this task is usually hindered by the inappropriate documentation of existing ontologies, notably their implicit assumptions.

Encoding: representing the conceptualization in some formal language, eg frames, object models or logic.

Documentation: informal and formal complete definitions, assumptions and examples (essential to promote the appropriate use and reuse of an ontology). Documentation is important for defining the exact meaning of terms within the ontology.

Evaluation: determining the appropriateness of an ontology for its intended application, including determining the consistency, completeness and conciseness of an ontology [14]. Conciseness implies an absence of redundancy in the definitions of an ontology and an appropriate granularity. For example, an ontology that modeled protein molecules at the atomic resolution when the amino acid level would suffice would not be considered concise.

1.2.5 Proof, trust and security

If the Semantic Web is indeed to become a global database, and if its development is evolutionary and distributed, then there are issues of accessibility, trust and credibility. Not all data sources will have universal access nor they will be equally reliable, so there needs to be a robust and extensible security model.

If instead of just returning an answer to a query, a Semantic Web application could also attach a proof of how that answer was derived, then the querying application could potentially do some reasoning about how *'believable'* that fact is. At the very least, derived facts could be attributed to a source, and over time applications could be developed which rate sources as to their integrity, etc.

These upper layers of the stack are the least researched and present some of the most difficult technical challenges faced by the Semantic Web venture.

1.3 Data Integration

Data integration is the problem of providing unified and transparent access to a collection of data stored in multiple, autonomous, and heterogeneous data sources [50], [65]. In formulating the queries, the user is freed from the knowledge on where data are, how data are structured at the sources, and how data are to be merged and reconciled to fit into the global schema.

The interest for data integration systems has been continuously growing in the last decade. The recent developments of Computer and Telecommunication technology, such as the expansion of the Internet and the World Wide Web, have provided the users with a huge number of information sources, generally autonomous, heterogeneous and widely distributed.

As a consequence information integration has emerged as a crucial issue in many application domains, e.g. distributed databases, data warehousing, data mining, data exchange, as well as in accessing distributed data over the web.

1.3.1 Theoretical Aspects

Different approaches to data integration are possible; they lead to different architectures and are based on different principles.

In particular, the integration approaches used in the existing systems can be classified first in terms of the data model they use - text, structured data or linked records. For systems that view sources as exporting mainly text, integration involves supporting keyword/text search across the sources. When the sources are viewed as exporting more structured data, there are two broad types of integration approaches, based on whether the data from the sources are *'warehoused'* or *'accessed on demand'* from the sources. Finally, for systems that view sources as exporting linked sets of browsable records, integration involves supporting effective navigation across sources.

Since the majority of systems use the (semi-)structured or linked record models [80], in the next sections we will limit the analysis of the different integration approaches to those two. In particular, we will deserve more attention to the mediator-based architecture, being it the core of the system described in this work.

1.3.2 Navigational Approaches

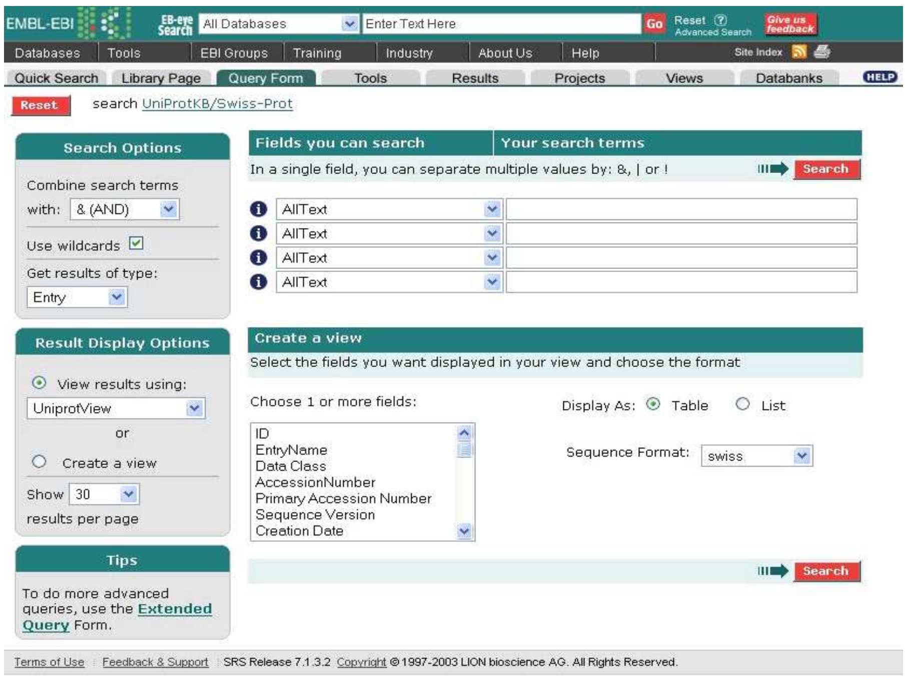
The idea behind navigational or link-based integration emerged from the fact that an increasing number of sources on the web ask the users to manually browse through several web pages and data sources in order to obtain the desired information [17].

In practice, queries are transformed into (several) path expressions that could each answer the query with different levels of satisfaction [55].

Navigational integration eliminates relational modeling of the data and instead applies a model where sources are defined as sets of pages with their interconnections and specific entry-points, as well as additional information such as content, path constraints, and optional or mandatory input parameters [62], [65].

Provided that multiple physical paths may link two sources, recent studies are trying to determine how to identify the best of several potential execution paths [72].

In fig.1.4 you can see a sketch of the SRS interface, the most representative example of a navigational integration system in the field of bioinformatics. Note how the SRS interface is similar to those of classical keyword-based information retrieval systems (search engines).



search UniProtKB/Swiss-Prot

Search Options

Combine search terms with: & (AND)

Use wildcards ☑

Get results of type: Entry

Result Display Options

[✓ View results using:] UniprotView

or

Create a view

Show 30 results per page

Tips

To do more advanced queries, use the Extended Query Form.

Fields you can search | Your search terms

In a single field, you can separate multiple values by: &, | or !

AllText
AllText
AllText
AllText

Create a view

Select the fields you want displayed in your view and choose the format

Choose 1 or more fields: ID, EntryName, Data Class, AccessionNumber, Primary Accession Number, Sequence Version, Creation Date

Display As: [✓ Table] List

Sequence Format: swiss

Terms of Use | Feedback & Support | SRS Release 7.1.3.2 Copyright © 1997-2003 LION bioscience AG. All Rights Reserved.

Figure 1.4: SRS Interface

1.3.3 Warehousing Approaches

Warehouse integration consists in materializing the data from multiple sources into a local warehouse and executing all queries on the data contained in the warehouse. Warehousing emphasizes data translation, as opposed to query translation in mediator-based integration [58].

Relying less on the network to access the data obviously helps in eliminating various problems such as network bottlenecks, low response times, and the occasional unavailability of sources. Furthermore, using materialized warehouses allows for an improved efficiency of query optimization as it can be performed locally [17].

This approach however has an important and costly drawback in terms of result reliability and overall system maintenance caused by the possibility of returning outdated results. Warehouse integration systems must indeed regularly check throughout the underlying sources for new or updated data and then reflect those modifications on the local copy of the data [17].

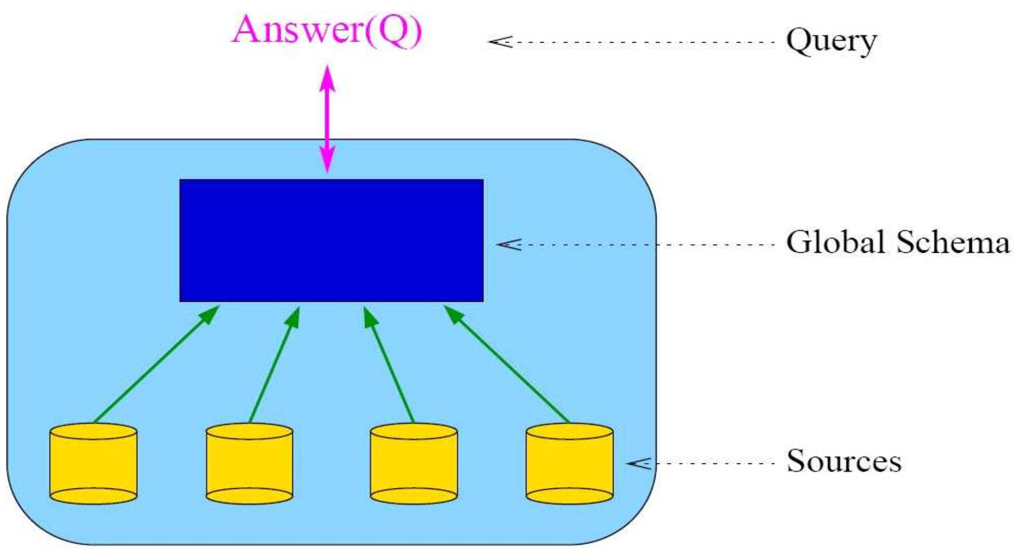


Figure 1.5: Conceptual model of a mediation system

1.3.4 Mediator-Based Approaches

In fig.1.5 you can see a logical integration schema, that is valid for both a warehousing and a mediator-based system (with the important difference that for a data warehouse the mappings - represented as green arrows in the picture - are to be interpreted as *data mappings*, used only while loading the global relational schema, while for a mediator-based schema the mappings are *schema mappings*, used during query processing to relate the global and the sources' schema).

Mediator-based integration concentrates on query translation. A mediator in the information integration context is a system that is responsible for reformulating at runtime an input query edited by a user over a global schema into a query on the local schema of the integrated data sources.

Many different conceptual models may be used as global schema, and vary from simple (object-)relational schema to more complicated and versatile ontologies (more on that in the next section).

Unlike in the warehouse approach, none of the data in a mediator-based integration system is converted to a unique format according to a data translation mapping. Instead a different mapping is required to capture the relationship between the source schema and the global schema, thus allowing queries on the mediator to be translated to queries on the data sources.

Specifying these mappings is a critical step in creating a mediator, as it influences both how difficult the query reformulation is and how easily new sources can be added to or removed from the integration system (or, more in general, updated).

Formally, a mediator-based data integration system I, is a triple $\langle G, S, M \rangle$ [65], [81], where:

- G is the global schema, i.e., a set of global relational symbols, each one with an associated arity (the number of its attributes), plus a set of integrity constraints expressed over such relational symbols;

- S is the source schema, i.e., a set of relational symbols (disjoint from G), that constitutes a relational representation of the data stored at the sources;
- M is the mapping between G and S, constituted by a set of assertions of the form $\{q_S, q_G\}$, in which q_S is a conjunctive query over the sources' schema, while q_G is a conjunctive query over the global schema.

Designing such a kind of data integration system is a very complex task, and is characterized by a number of issues (most of which are in common with the designing process of different systems), including:

1. modeling the system, i.e. defining both the global schema and the relationships (mappings) between the global schema and the sources;
2. dealing with incomplete data sources;
3. dealing with inconsistent data sources;
4. dealing with limitations on accessing the sources;

In particular, with respect to the mapping assertions different assumptions can be done, that affect the notion of satisfaction of mapping. In general, if we assume that the mapping is sound, then we have that the data provided by the sources are a subset of the global data - the extension of q_S is contained into the extension of q_G. Conversely, if the mapping is considered to be complete the data provided by the sources are a superset of the global data - the extension of q_S contains the extension of q_G. Finally we say that a mapping is exact, when it is both sound and complete. It should be pointed out that, due to the general characteristics of the sources, that are distributed, autonomous and independent, the sound mapping assumption is more reasonable in a data integration environments [65].

The mapping design is one of the crucial tasks in defining a data integration system specification. In fact different representations with different and well-known proprieties can be obtained. We say that a mapping assertion follows the global-as-view (GAV) paradigm, when q_G corresponds to a full query over a single global relation: an assertion of that kind gives a straightforward specification of the global data, in terms of the source data. Dually, the local-as-view (LAV) approach, let us define q_S as a full query over a single source relation.

Both formalisms present advantages and drawbacks [23]. GAV mappings ease the query answering process that can be done by means of simple unfolding, but its structure is not well suited for updating the sources: every change in the source schema may lead to redesigning the mapping assertions. LAV mappings instead are well suited for that, because adding or removing a source specification only involves adding or removing a single mapping assertion. On the other hand, query answering in LAV is hard, providing the mapping views only partial information about the elements of the global schema.

In a nutshell LAV is considered to be much more appropriate for large scale ad-hoc integration because of the low impact changes to the information sources have on the system maintenance, while GAV is preferred when the set of sources being integrated is known and stable [80].

Example 1.1 Let us consider the following scenario, where we have the relational global schema:

Global schema:

movie (Title, Year, Director),

european (Director),

review (Title, Critique),

and there are three source relations:

Source 1: s_1 (Title, Year, Director), since 1960, european directors,

Source 2: s_2 (Title, Critique), reviews of movies with european director,

Source 3: s_3 (Director), european directors.

The GAV mappings associate to every relation of the global schema a view over the sources' schema:

$$\begin{gathered} \textit{movie}\ (\mathrm{X, Y, Z}) \longleftarrow s_1\ (\mathrm{X, Y, Z}), \\ \textit{european}\ (\mathrm{X}) \longleftarrow s_3\ (\mathrm{X}), \\ \textit{review}\ (\mathrm{X, Y}) \longleftarrow s_2\ (\mathrm{X, Y}), \end{gathered}$$

while the LAV mappings are given by (using conjunctive queries with arithmetic comparisons [15]):

$$\begin{gathered} s_1\ (\mathrm{X, Y, Z}) \longleftarrow \textit{movie}\ (\mathrm{X, Y, Z}),\ (\mathrm{Y} \geq 1960), \\ s_2\ (\mathrm{X, W}) \longleftarrow \textit{movie}\ (\mathrm{X, Y, Z}),\ \textit{review}\ (\mathrm{X, W}),\ \textit{european}\ (\mathrm{Z}), \\ s_3\ (\mathrm{X}) \longleftarrow \textit{european}\ (\mathrm{X}). \end{gathered}$$

●

A recent approach (called GLAV) tries to generalize the LAV approach [32], [70].

For what concerns the problems of incomplete and potentially inconsistent (and not fully accessible) sources, they are common to all the integration approaches we presented, and can be considered separate hot research topics. For this reason we will deserve particular attention to them, in one of the following sections. A good (and more detailed) theoretical and practical treatment of these problems is provided by Lembo [81].

Ontology-driven Data Mediation

A mediated schema is called *ontology-driven* (or, that is the same, *ontology-based*) when its global schema is an ontology.

Many authors have proposed the use of ontologies for integrating heterogeneous sources, although the approach presents several challenges [33], [38].

The development of a single schema or ontology is a serious and expensive task, best tackled as a joint exercise with domain experts, by merging and adopting pre-existing ontologies, with the intention that the result will be reusable by other applications. This task is difficult [42].

The use of a single terminology by a mediator requires that the user know what is in the terminology, understand what the terms and concepts mean, and buy into it. Gaining consensus is particularly difficult because one user's or community's vocabulary might differ from that of another. The ontology will need to be tended and updated to cater to new sources or changes in sources. It also needs to be comprehensive enough to cater to an appropriately adequate range of resource types.

Interpretations of concepts often depend on context, and one ontology cannot be viewed as a repository of all possible interpretations [33], [42]. Attempts to tackle this issue range from the adoption of de facto common vocabularies by a community prepared to adapt to some form of common consensus (for example, the Gene Ontology [35]), to mechanisms for defining ontological commitment, multiple definitions for concepts in the same ontology, and ontological views.

In the past these problems have often hindered the practical exploitation of knowledge-based information integration systems in many challenging disciplines, basically preventing researchers from developing suitable domain ontologies. As for the biological domain, bioinformatics researchers have recognized that semantic schema and data matching could be aided by a comprehensive thesaurus of terms or a reusable reference ontology of biological concepts [17], [19].

Furthermore the querying capabilities deriving from the use of an ontology as global schema are much more powerful than those provided by the adoption of simple (object-)relational schema, and are at the same time absolutely necessary to answer the need for data mining that is typical of this research field.

Query Processing

The problem of query processing is concerned with one of the most important issues in a data integration system, that is the choice of the method for computing the answer to queries posed in terms of the virtual global schema only on the basis of the data residing at the sources [81]. The main issue is that the system should be able to re-express such queries in terms of a suitable set of queries posed to the sources, hand them to the sources, and assemble the results into the final answer.

It is worth underlining that, while for a GAV-based integration system, query processing is essentially equivalent to unfolding over the sources' schema the query written on the global schema, with a LAV mapping approach things are much more complicate.

In fact, in LAV the views in the mapping provide in general only a partial knowledge about the data that satisfy the global schema, hence query processing is inherently a form of reasoning in the presence of incomplete information [4], [24].

In other words, in GAV the mapping essentially specifies a single database for the global schema, hence evaluating the query over this database is equivalent to evaluating its unfolding over the sources. On the contrary, since in LAV several possibilities of populating the global schema with respect to the source extensions may exist, the semantics of a LAV system has to be given in terms of several database instances for the global schema, which have to be taken into account in processing the user query.

In the LAV approach query processing has been traditionally solved by means of query rewriting, that is performed in two steps: in the first step the query is reformulated in terms of the views (that are the sources' schema), and in the second the obtained query is evaluated on the view extensions, i.e. a database instance for the source schema.

Example 1.2 Let us consider the same scenario of Example 1.1, and suppose the user poses the following query:

$$\text{q (X, Z)} \longleftarrow \textit{movie}\ \text{(X, 1998, Y)},\ \textit{review}\ \text{(X, Z)}$$

asking for title and reviews of movies produced in 1998. in the GAV case, the answer is computed by simple unfolding, resulting in the following query over the sources:

$$\text{q (X, Z)} \longleftarrow s_1 \text{ (X, 1998, Y)}, s_2 \text{ (X, Z)}$$

that can be directly evaluated over the sources. In the LAV case, a rewriting of the query q is:

$$q_r \text{ (X, Z)} \longleftarrow s_1 \text{ (X, 1998, Y)}, s_2 \text{ (X, Z)}$$

such a rewriting can be evaluated over the sources. Its unfolding according to the mapping assertions given in Example 1.1 produces the query over the global schema:

$$q_r\text{(T, R)} \longleftarrow \textit{movie}\text{(T, 1998, D)}, \textit{movie}\text{(T, Y, D')}, \textit{review}\text{(T, R)}, \textit{european}\text{(D')}$$

where we did not write the atom 1998 $\geq$ 1960, which is clearly true.

●

A different approach to LAV query processing, more general than query rewriting, consists in not posing any limitations on how the query is going to be processed: all possible information, in particular the view extensions, can be used for computing the answers to the query. This approach is commonly called *query answering.* We point out that the ultimate goal of query answering is to provide the certain answers to a user query, that is to compute the intersection of the answer sets obtained by evaluating the query over any database that satisfies the global schema.

1.3.5 Incompleteness and Inconsistency

Incomplete Data

As we said above, query processing in LAV could be considered a form of reasoning in presence of incomplete information. Hence, sources in LAV data integration systems are generally assumed to be sound, but not necessarily complete (i.e. each source concept is assumed to store only a subset of the data that satisfy the corresponding view on the global schema). A different approach is followed for processing queries in GAV, where the form of the mapping allows for the direct computation of a global database instance over which the user queries can be evaluated (unfolding).

Note that sometimes also in GAV systems the sources provide only a subset of the data that satisfy the global schema, hence views in the mapping should be considered sound rather than exact. This becomes particularly relevant when integrity constraints are specified on the global schema [78].

Hence, in the presence of incomplete data with respect to integrity constraints specified on the global schema, unfolding is in general not sufficient to answer a user query in GAV, and reasoning on the constraints is needed in order to compute the certain answers to the query (clearly the same is true in the LAV case).

Inconsistent Data

Let us consider the case of a relational integration system with a sound mapping. Let us suppose that a key constraint is violated on the relational global schema: the soundness assumption on the mapping does not allow us to disregard tuples with duplicate keys, hence the data are inconsistent with respect such constraint.

This is a common situation in data integration, since integrity constraints are not related to the underlying data sources, but rather to the semantics of the global schema (or, that is the same, to the real world). That is why we cannot expect independent and autonomous data sources to produce data which obey those constraints. On the other hand, since most of the data could satisfy such constraints, it seems unreasonable to consider the entire system inconsistent [81].

Classical assumptions on the views do not allow to properly handle data inconsistency, since they generally lead to a situation in which no global database exists that satisfy both the integrity constraints and the assumption on the mapping, and it is not possible to provide meaningful answers to user queries.

A possible solution to this problem is to characterize the semantics of a data integration system in terms of those databases that satisfy the integrity constraints on the global schema, and approximate *'at best'* the satisfaction of the assumptions on the mapping, i.e. in a way that is *as close as possible* to the interpretation of the mapping.

More on this difficult and broad topic can be found in [15], [30], [48], [61], [69], [81].

1.4 Molecular Biology and Bioinformatics

It is undeniable that, among the sciences, life science and in particular biology played a key role in the twentieth century. That role is likely to acquire further importance in the years to come. In the wake of the work of Watson and Crick [75], and the sequencing of the human genome, far-reaching discoveries are constantly being made.

The enormous amount of data gathered by biologists, and the need to interpret it, requires tools that are in the realm of computer science. This need led in the last two decades to the birth of the interdisciplinary science called *bioinformatics*.

A distinctive aspect of bioinformatics is its widespread use of the Web. The immense databases containing DNA sequences and 3D protein structures are available on-line to almost any researcher. Furthermore, the community interested in bioinformatics has developed a myriad of application programs accessible through the Internet. Some of these programs (e.g., BLAST) have taken years of development and have been finely tuned. The vast numbers of daily visits to some of the NIH sites containing genomic databases are comparable to those of widely used search engines or active software downloading sites.

In the following sections, we will present some of the main issues in the field of bioinformatics, trying to make the reader understand or at least get a closer idea of the amount of data bioinformaticians have to manage for their researches.

Genome Sequencing

Each cell of a living organism contains chromosomes composed of a sequence of DNA base pairs. This sequence, the *genome*, represents a set of instructions that controls the replication and function of each organism.

The automated DNA sequencer gave birth to genomics, the analytic and comparative study of genomes, by allowing scientists to decode entire genomes. Although genomes vary in size from millions of nucleotides in bacteria to billions of nucleotides in humans and most animals and plants, the chemical reactions researchers use to decode the DNA base pairs are accurate for only about 600 to 700 nucleotides at a time.

The process of sequencing begins by physically breaking the DNA into millions of random fragments, which are then "read" by a DNA sequencing machine. Next, a computer program called an *assembler* pieces together the many overlapping reads and reconstructs the original sequence.

These techniques have been largely improved in the last two decades [67], and researchers have currently access to several complete new genomes every year.

Protein Structure Prediction

With the rapid growth of the number of yearly completely sequenced genomes, the post-genomic problem of gene function identification has become more pressing with time. Predicting the structures of proteins encoded by genes of interest is one possible means to glean subtle clues as to the functions of these proteins [40].

Thus, the research field of protein structure prediction has seen in the last decade the proposition of a plethora of different methods and algorithms addressing the issue, and the birth of international committee for the evaluation of such algorithms (CASP [91]).

Early work in the structure modeling field primarily focused on understanding the nature of the natural folding process and on the development of physics-based force fields to determine the relative free energy of any conformation of a polypeptide chain.

These methods have largely been supplanted by more successful *'knowledge-based'* approaches, which utilize the large and rapidly growing number of experimentally determined structures and sequences in a variety of ways. As a consequence, the accuracy of models depends on similarity to already known structures.

Again, the multitude of data to be managed and analyzed to perform these tasks is overwhelming and impossible to be manually processed and mined.

Evolutionary Biology

Evolutionary biology is founded on the concept that organisms share a common origin and have subsequently diverged through time. Phylogenies (typically formulated as trees) represent our attempts to reconstruct evolutionary history. Phylogenetic analysis is used in all branches of biology with applications ranging from studies on the origin of human populations to investigations of the transmission patterns of

HIV [66], and beyond, with a variety of uses in drug discovery, forensics, and security [43].

The accurate estimation of evolutionary trees is a challenging computational problem. For a given set of organisms (or *taxa*), the number of possible evolutionary trees is exponential [76]. An exhaustive search through the tree space is certainly not an option. Thus, scientists have designed a plethora of heuristics to assist them with phylogenetic analysis.

But even with the application of these heuristics, the quantity of data obtained from real case-studies is overwhelming, so effective and efficient tools for data management are a 'must' for a profitable mining activity.

1.4.1 Ontologies in Biology

As we previously said (sec. 1.2.4), Ontologies are used for communication between people and organizations by providing a common terminology over a domain. They provide the basis for interoperability between systems. They can be used for making the content in information sources explicit and serve as an index to a repository of information. Furthermore they can be used as a basis for integration of information sources and as a query model for information sources. They are being used nowadays in many areas, including bioinformatics.

Biologists need knowledge to perform their work, often using a pre-existing item of knowledge to make inferences about the item under investigation. This is why it is sometimes said that biology is a *'knowledge-based'*, rather than an *'axiom-based'* discipline [25].

Modern biologists also need knowledge for communication. Biology is a data-rich discipline, which is available as a fund of knowledge by which biologists generate further knowledge. This knowledge is stored in thousands of databases, many of which need to be used in concert during an investigation. Knowledge is vital in two respects during this process. First, when using more than one data store or analysis tool, a biologist needs to be sure that knowledge within one resource can be reliably compared with another, i.e. knowledge is necessary to integrate information from different sources (see [38]).

The second need for knowledge is to define and constrain data within a resource. Biological data can be very complex; not only in the type of data stored, but in the richness and constraints working upon relationships between those data. When designing a database it is useful to be able to describe what values can be specified for which attributes under which conditions. This is the encapsulation of biological knowledge within database schema.

It is impossible for one biologist to deal with all the knowledge within even one sub-domain of their discipline. The continuous arrival of whole genomes and the knowledge they contain only exacerbates the situation.

The need for systems that can apply the domain experts' knowledge to biological data, or at least can help experts apply it, poses numerous questions, in particular regarding how knowledge can be captured to make it available and useful within computer applications.

Well, Knowledge can be captured and made available to both machines and humans by an ontology. A common ideal for an ontology is that it should be re-usable [12]. This ambition distinguishes an ontology from a database schema, even though both are conceptualizations. In fact, a database schema is intended to satisfy only one application, while an ontology could be reused in many applications. However an ontology is only reusable when it is to be used for the same purpose for which it was developed. Not all ontologies have the same intended purpose and may have parts that are reusable and other parts that are not. They will also vary in their coverage and level of detail.

We can divide ontology use into three broad categories:

- *Domain-oriented*, which are either domain specific (e.g., Escherichia Coli) or domain generalizations (e.g., gene function or ribosomes).
- *Task-oriented*, which are either task specific (e.g., annotation analysis) or task generalizations (e.g., problem solving).
- *Generic*, which capture common high-level concepts, such as Physical, Abstract, Structure and Substance. This can be especially useful when trying to reuse an ontology, as it allows concepts to be correctly or more reliably placed. It can also be important when generating or analyzing natural language expressions using an ontology. Generic ontologies are also known as ‘upper ontologies’, ‘core ontologies’ or ‘reference ontologies’.

Most bio-ontologies have a mixture of all three types of ontology. A well-formed ontology will be built in a modular way using a mixture of generic domain, generic task and application ontologies. Its parts will be clearly defined so that they can be reused. A less well-formed ontology will have blurred distinctions, making reuse and modification harder [42].

Other measures for the quality of an ontology include its *clarity*, *consistency*, *completeness* and *conciseness* [12].

In the remaining part of the section a representative small sample of existing bio-ontologies will be shortly reviewed (see [42] for a more extensive and exhaustive survey):

- The *RiboWeb* Ontology.
- The *Gene* Ontology (GO).
- The *TAMBIS* Ontology (TaO).

The RiboWeb Ontology

RiboWeb [29] primarily aims to facilitate the construction of 3D models of ribosomal components and to compare the results to existing studies. The knowledge RiboWeb uses to perform these tasks is captured in four ontologies: the physical-thing ontology; the data ontology; the publication ontology and the methods ontology.

The physical-thing ontology describes ribosomal components and associated *'physical things'*. The data ontology captures knowledge about experimental detail as well as data on the structure of physical-things. The methods ontology contains information about techniques for analyzing data. It holds knowledge of which techniques can be applied to which data, as well as the inputs and outputs of each method.

The constraints described within RiboWeb can highlight conflicts with current knowledge to the biologist.

The Gene Ontology (GO)

The Gene Ontology Consortium [39] is a joint project. The project's goal is to produce a structured, precisely defined, common and dynamic controlled vocabulary that describes the roles of genes and proteins in all organisms (Gene Ontology Consortium, 2000).

Currently, there are three independent ontologies publicly available over the Internet: *biological process*, *molecular function* and *cellular component.* The biological process ontology deals with biological objectives to which the gene or gene product contribute. A process is accomplished via one or more ordered assemblies of molecular functions. The molecular function ontology deals with the biochemical activities of a gene product. It only describes what is done without specifying where or when the event takes place. The cellular component ontology describes the places where a gene product can be active. The GO ontologies are becoming a de facto standard and many different bio-databases are today annotated with GO terms [73]. The ontologies grow continuously. The terms in GO are arranged as nodes in a directed acyclic graph, where multiple inheritance is allowed. The two most important relations that are modeled are the is-a relation and the part-of relation.

The TAMBIS Ontology (TaO)

The TAMBIS Ontology (TaO) [47], [31] describes a wide range of bioinformatics tasks and resources, and has a central role within the TAMBIS data integration system.

An interesting difference between the TaO and the greatest part of the other ontologies is that the TaO does not contain any instances. The TaO only contains knowledge about bioinformatics and molecular biology concepts and their relationships, the instances they represent still reside in the external databases. As concepts represent collections of instances, a concept can act as a question.

The concept *Receptor Protein*, for example, represents the instances of proteins with a receptor function and gathering these instances is answering that question.

The TaO is a dynamic ontology, - it can grow without the need for either conceptualizing or encoding new knowledge. In contrast, the other ontologies described, are static - developers must intervene and encode new conceptualization to form new concepts.

The TaO is available in two forms, a small model that concentrates on proteins and a larger-scale model that includes nucleic acids. The small TaO, with 250 concepts and 60 relationships, describes proteins and enzymes, as well as their motifs, secondary and tertiary structure, functions and processes.

The larger model, with 1,500 concepts, broadens these parts to include concepts pertinent to nucleic acid, its children and genes.

1.4.2 Integration Tools for Molecular Biology

The integration of biological data is just one phase of the entire molecular biology research and genomic hypothesis discovery process. However the use of non-manual techniques (i.e., computers) in the knowledge integration process has never been felt as an actual need by biologists, even when the task resulted to be extremely time consuming.

Now that relevant data are widely distributed over the Internet and made available in different formats, manual integration has become practically infeasible. The amount of data stored in biological databases has indeed grown exponentially over the past decade [94], while simultaneously the number of available biomolecular and genomic sources on the web has increased to more than 500 [56], [68].

Furthermore, the need for effective integration of bioinformatic sources is also justified by the characteristics of these sources [80]:

- the highly diverse nature of the data stored;
- the representational heterogeneity of the data;
- the autonomous and web-based character of the sources and the way the data is published and made available to the public;
- the various interfaces and querying capabilities offered by the different sources.

In addition to the traditional issues characterizing general data integration problems, the particular nature of the data and the particular attention you need to pay in such a delicate research field add some other complications and challenges to be resolved:

- **Variety of Data**: the data exported by the available sources cover several biological and genomic research fields. Furthermore, bioinformatic data can be characterized by many relationships between objects and concepts, which are difficult to identify formally. Finally, not only can the quantity of data available in a source be quite large, but also the size of each datum or record can itself be extremely large.
- **Autonomous and Web-based Sources**: most of these sources operate autonomously, which means that they are free to modify their design and/or schema, remove some data without any prior "public" notification, or occasionally block access to the source for maintenance or other purposes. Furthermore, new discoveries or experiments will continually modify the source content to reflect the new hypotheses or findings. In fact the only way for an integration system to be certain that it will return the latest data is to actually access the sources at query time [80].
- **Access Limitation**: the sources often allow for only certain types of queries to be asked, thereby protecting and preventing direct access to their data. These intentional access restrictions force end-users and external systems to adapt and limit their queries to a certain form [58].

In the remaining part of the chapter, we will present some relevant data integration systems for the molecular biology research.

SRS

The Sequence Retrieval System (SRS) is closer to a keyword-based retrieval system than an integration system. Its approach to bioinformatic integration is to parse flat files or databanks that contain structured text with field names. It then creates and stores an index for each field and uses these local indexes at query-time to retrieve relevant entries [52], [56].

Although extensive indexed entries are kept locally to be used by the query processor at query time, SRS is not actually a warehouse system as the actual data is neither modified nor stored locally. The other main feature of SRS is that it keeps track of the cross-references between sources.

The results of a query in this system are essentially composed of a set of tuples or entries directly retrieved from initially selected sources, and a set of paths across other sources which lead to information that is related to the query. Thus, a user can browse through a set of sources in a point-and-click type of navigation [17] even after having submitted a very simple query, and find more relevant or complementary results in the suggested links.

BioKleisli

BioKleisli is primarily a loosely-coupled federated database system. The mediator on top of the underlying sources relies mainly on a high-level query language that is more expressive than SQL and that provides the ability to query across several sources: the Collection Programming Language, or CPL [16], [45].

The data model used in BioKleisli is an object-oriented type system that is more expressive than the relational model since it includes bags, lists, variants, nested sets and nested records. BioKleisli does not use any global molecular biology schema or ontology that the user could use to formulate queries. This approach therefore requires of the users not only a strong competency in CPL [45] but also a perfect knowledge of the schema and structure of the bioinformatic sources being integrated.

The BioKleisli project is mainly aimed at performing a horizontal integration. Furthermore, no optimization based on source characteristics or source content is performed.

TAMBIS

TAMBIS (Transparent Access to Multiple Bioinformatics Information Sources) is a mediator-based and ontology-driven integration system [25], [34], [46].

Queries in TAMBIS are formulated through a graphical interface where a user needs to browse through concepts defined in a global schema and select the ones that are of interest for the particular query. Because TAMBIS needs external wrappers, it uses wrappers from the BioKleisli system to access the underlying sources. The planning and optimization subsystem in TAMBIS only performs reordering of query components; it does not store source statistics or analyze source capabilities.

It is important to note that the ontology defined by TAMBIS is not primarily used for schema mapping between the underlying bioinformatic sources; instead the ontology is a dictionary and classification of biological concepts representing subsumption relationships between concepts. The mapping of ontology concepts to source-dependent CPL functions is done by another subsystem called the Source Model which simply captures which CPL function is related to which ontology concept. Hence the TAMBIS domain ontology mainly serves the purpose of easing the user's task of formulating the query.

"Good ideas are not adopted automatically. They
Must be driven into practice with courageous patience."

Hyman Rickover.

Chapter 2

Modeling the System

In this chapter, we will present in details the architectural and the modeling choices that characterized the design of the integration system. In particular, we will first present the theoretical background at the basis of our choices, then we will focus on the mediation module, that represents the core of the integration architecture.

Additionally, at the end of the chapter an explanation of the algorithms of query processing and of the schema mappings generation process is presented.

2.1 System Architecture

The continuous update of the sources being integrated (and the birth of brand new ones), the periodical changes of the format of the data stored, and the consequent need for a quick extension of the system to support novel functionalities and query capabilities (and to integrate new data banks) led to the design of a modular architecture.

In particular, we adopt a mediator based architecture, that we think to be the more suitable to future updates when compared to a warehousing approach; and that is better suited to analyzing and mining data than a navigational system, which is usually very close to a keyword-based search engine (see sec. 1.3.1).

We exploit a LAV approach to map the sources, i.e. every source schema can be seen as contained in the results of a query over the global schema (in other words: the sources are views, and we have to answer queries on the basis of available data in the views).

To the aim of representing the knowledge of the domain and mediate through the different data sources, we developed a model ontology in RDF. The choice of using RDF instead of OWL or other ontology representation languages, not lacking a reasoning engine, is motivated by the fact that we do not really want to make any reasoning on the information stored in the ontology, but only to provide the user with simple query capabilities (i.e., the same provided by the integrated data sources), and a flexible and readable representation (and RDF is probably the best choice for that), while focusing the data mining and all the analytical processes on the data retrieved from the data sources.

Also note that our ontology is lacking the extensional part, the data that represent all our possible instances residing at the source level. Our ontology is meant only to represent the concepts and resolve the semantic heterogeneities for the user, and it is only exploited for editing queries (from the user's perspective), and for dividing the queries in source-dependent queries (from the system's perspective).

A detailed sketch of the system architecture is depicted in fig.2.1.

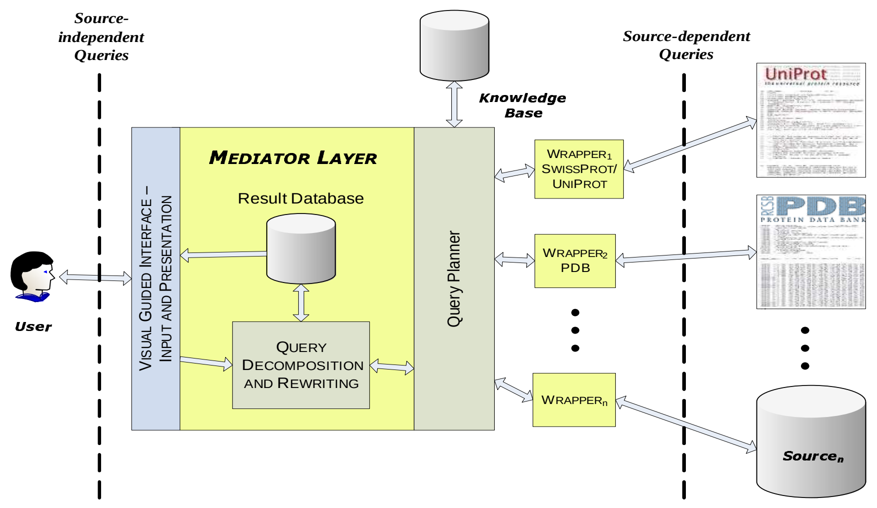

Source-independent Queries
Knowledge Base
Source-dependent Queries
MEDIATOR LAYER
Result Database
VISUAL GUIDED INTERFACE – INPUT AND PRESENTATION
User
QUERY DECOMPOSITION AND REWRITING
Query Planner
WRAPPER1 SWISSPROT/ UNIPROT
WRAPPER2 PDB
WRAPPERn
UniProt
PDB
Sourcen

Figure 2.1: System Architecture

The user communicates with the system through a visual interface, which provides a guided input mechanism and a result-browsing tool. The (source-independent) query provided by the user is analyzed by the mediator, which opportunely compiles it into a set of (source-dependent) queries to be submitted by wrappers to the integrated sources at execution time.

Generated queries are then passed to a query planner, which is responsible of establishing the query execution order and providing the wrappers with intermediate results, where needed.

Finally, wrappers query the remote sources to get the requested information, and return them to the mediator for further analysis and presentation.

As an example, consider the case in which the user submits a query like: $Q =$ *"Find the sequence S of all the proteins P with a given structural motif M"*. The mediator module knows from the stored mappings that information concerning protein sequences are to be retrieved from a certain set of sources, S_1; while information concerning protein structural motifs are contained in another set of sources, S_2 (note that the two sets may be overlapping). Hence the mediator finds which attributes (concepts in the ontology) are *common* to the sources in S_1 and S_2 (or, even better, if there is a cross-reference between the sources); let us consider the worst case of non-referencing sources, and let us say the sources share a single attribute (for sake of simplicity), which is called *'name'*.

Note that some common attributes (ID, Date, etc.) are excluded a priori from this process, because the instances representing the same object may have different values for it[1].

Then the mediator divides the query into two elementary (in the sense they involve only a selected attribute - e.g. name - and a constrained one - e.g. motif) sub-queries of the kind: $SQ_1 =$ *"Find the name N of all proteins with structural motif M"*, and

[1]Actually we even put in discussion whether the representation of these attributes/concepts was needed at all; we resolved to represent them so not to limit the querying capabilities of the system, even if our choice complicated a bit the query processing, introducing the problem of data consistency (see sec. 1.3.5).

$SQ_2 =$ *"Find the sequence S of all protein with name N"*.

Finally the mediator plans the execution (recognizing the second sub-query needs as input the results of the first sub-query), translates SQ_1 in a list of conditions on the source attributes, for each of the sources in S_2, and calls the suitable wrappers.

Each wrapper accesses the sub-query/list it has to process and executes it, retrieving the results (i.e. the desired *protein name*) at the source level and storing them in a local *result repository.*

The mediator accesses the results, and use them to perform the translation process on SQ_2, this time calling on the wrappers for the sources in S_1. Finally, the mediator analyzes the whole results to find possible inconsistencies (in that case, the mediator presents all the possible results to the user, specifying for each of them the provenance of the data, and a grade of reliability - based on the principle: *'the more frequent, the more reliable'*), then present the list of results, indicating for each of them the provenance (source) and the date of the last modification (where this information is readable at the source level), together with hyperlinks to the corresponding entries in the remote sources.

In the following section we theoretically motivate the design choices we made, then we present a detailed description of every component, underlying its functionality and the challenges we had to face to develop it.

2.2 Theoretical Background

2.2.1 Mediator-Based Data Integration

As previously stated we rely on a mediator-based architecture, the approach that we consider better suited to be applied to the biological domain.

In fact, the frequent updates of the data contained in the remote sources (and the possible introduction of brand new data banks or the removal of older ones) and the possible unavailability of the source schema make a warehousing approach in practice unfeasible.

On the other hand, the potential for mining and analyzing data, with the possibility of making (even simple) local inference on a unique global schema, and the annotation capability you get with warehousing and mediated approaches (something that is important and desirable in Bioinformatics) are totally absent in navigational systems, that are more similar to keyword based retrieval systems (see the SRS system [52] for example).

As we said above (see sec. 1.3), different assumptions can be done with respect to the mapping assertions, that affect the notion of satisfaction of mapping.

As a design choice, we don't actually implement neither a sound nor a complete mapping (as described in the previous chapter). In fact, we don't necessarily need to represent all the attributes constituting the schema of the sources being integrated, and on the other hand we may need information that are not present in the sources' schema. Hence, in our case source and global schema are in general simply overlapping. We may say that our mapping assertions are *'at most'* sound, in the sense that sources may be supposed to be full represented in the global schema (but that is not true at any given moment, provided that the sources may change their schema without any alert, and at the same time the user may modify the global schema), but some sources can be at some time points simply intersecting the global schema (until the global schema is extended to cover the missing attributes/relations).

For what concerns the mapping formalism we decided to adopt a LAV approach, that is a natural choice once observed that in the biological domain the sources can be subject to particularly frequent changes/updates.

In fact, exploiting a LAV paradigm, every time a (new) source is updated (added), we don't need to reassert the totality of the mappings (something that is in general true with a GAV approach), but simply to modify the (to add a) mapping for that source (see sec. 1.3.4).

2.3 Model Ontology

Given the choice of adopting a mediator-wrapper architecture, the first and more urgent issue we have to face is the generation of an internal representation of the information we are about to integrate, which could be easily updated (and accessed for browsing purposes) and mapped onto distributed (not always structured) repositories.

As we said above (see sec. 1.4.1) information integration requires a consistent shared understanding of the meaning of that information. The biologists' knowledge of molecular biology and bioinformatics, and their interpretation of the resources with respect to this knowledge, is essential to the task of combining resources to answer queries. A shared understanding requires three things: metadata, terminologies and ontologies.

In particular, ontologies provide a shared and common understanding of a domain that can be communicated across people and applications, and play a major role in supporting information exchange and discovery [38]. The resources may overlap in their content, but they certainly vary considerably on the view that is taken of that content, for example 'what is meant by *gene*?'. A comprehensive thesaurus of terms or a reusable reference ontology of biological concepts is a prerequisite for information integration [19].

The choice of using a domain ontology as global schema only move the problem of the information representation forward to the choice of an opportune representation language for the ontology, with all deriving consequences of this choice (as the capability of making inference and the computability of logic assertions).

The analysis of the possible solutions quickly reduced our alternatives to RDF and OWL (actually the lighter versions of the language, -lite and -DL, provided that the -Full version is equivalent to RDF). As previously stated (sec. 2.1), we eventually adopted an RDF representation on the basis of the important consideration that we do not need complex reasoning on our data, so there is no purpose in self-limiting the expressivity of the language.

After choosing the representational language, the most difficult aspects of the ontology generation process are choosing which concepts to represent, and collecting coherent and consistent descriptions of these concepts (and of the relationships between them).

While the former problem has been easily overcome via an accurate analysis of the sources, the latter required much more attention and efforts, and led us to the conclusion that only after a period of training and updates, on the basis of the user's view of the domain (that is to say *the user's personalization of the system*), the description of the concepts in the ontology could be considered accurate, precise and at the same time flexible enough for the system to properly satisfy user needs.

In fact, provided that the system should solve the representational and interpretational heterogeneities of data for the user, it is our belief that only the user can provide her/his intended meaning for them. The global ontology has thus to be viewed as an ever changing model, subject to refinement and enrichment during its entire life cycle. To this purpose, we are now working to provide the user interface with a tool to edit the model ontology, even if at the same we recognize it could be dangerous for the correctness of the global model to set the unexperienced user free to modify the global schema.

Note that although ontologies might seem to be abstract entities, it is possible to illustrate them as graphs in which vertexes (nodes, leaves) and edges (lines connecting the nodes) represent the terms and the rules of the ontology. For bio-ontologies, this graph is usually no more than a hierarchy [77]: this will be simple if each term has a single parent (such as in a taxonomy) and more complicated if a term has two or more parents or relationships (panel b). An example of the latter would be the *Gene Ontology (GO)* [39].

Our global ontology is not a hierarchy like the GO, but more properly a directed acyclic graph (with each node being a subject/object of an RDF triple - see Appendix A -, and each arc a relation between two nodes), i.e. where every relationship is directed, and it is not possible to make closed loops.

In the next sections, we might as well refer to the generated global ontology with the term *global graph*, to the represented entities with the term *node*, and to the relations with the term *arc*.

2.3.1 Building the Ontology

Given the extent of the molecular biology domain, the whole ontology on which the system should base its functioning is far from being completed, and probably the creation process would still take a couple of years of collaborative work with domain experts.

To the aim of testing the system we limited the generation process to only a small portion of the global ontology, developing it mainly *around* the concept of *'Protein'*.

It is worth pointing out that the entities and relations represented in the ontology are not necessarily represented in any data source related to the protein domain. In fact we decided to approach the creation of our ontology in such a way that we could represent all the available knowledge for the domain, independently from its actual use in the sources being integrated, so to possibly make future updates easier (if some data sources, on the basis of evidences from future experiments and studies, would decide to represent previously neglected characteristics of the information stored, we will maybe find ourselves a step ahead in the integration process, if those aspects were already considered in the model ontology).

To build the ontology we followed the general steps we outlined in sec. 1.2.4. A sketch of a portion of the global ontology is depicted in fig.2.2.

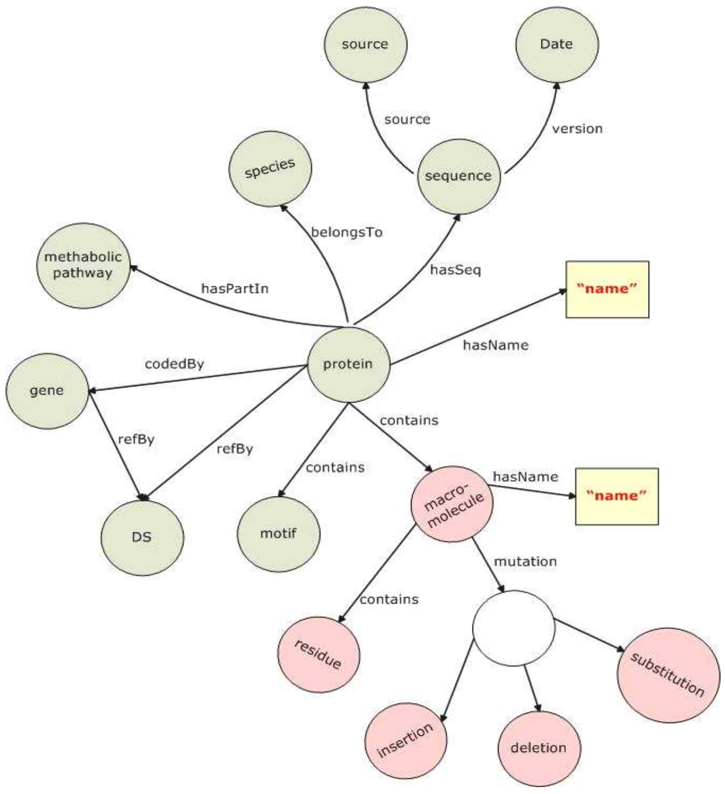


Figure 2.2: BioInView global ontology - portion

Note that, as it is clear for the concepts *'Protein'* and *'Gene'*, each concept in the ontology with a correspondent in one or more data sources has a reference to the data source/s.

These references are generated during the mapping process, and are used to ease the query decomposition task.

2.4 Editing and Browsing Interface

The most important modules of the user graphic interface are the query editor and the result browsing modules. The former is still under development and it is going to be basically a wizard guiding the user in writing correct queries (with no need of correctness checking mechanisms); the latter is a bit more complex than the result page of traditional search engines, where summaries of the results of the query are listed and links to the extended version of the retrieved entries are provided.

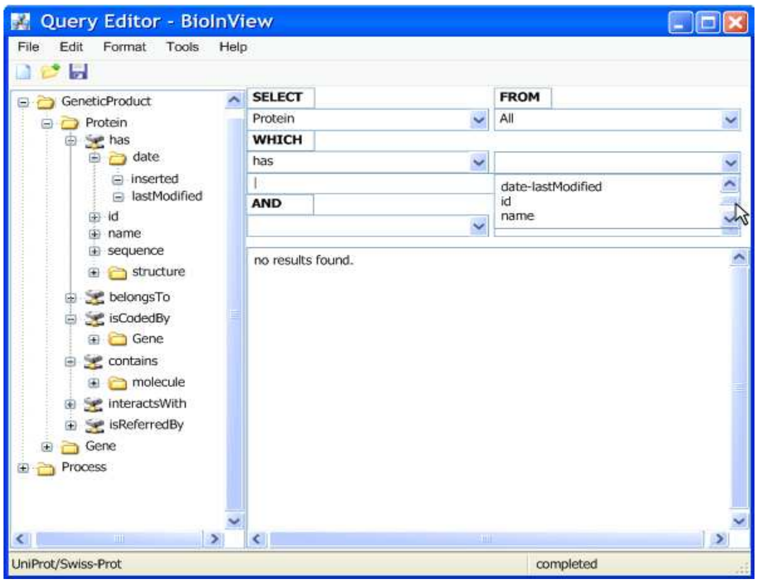
2.4.1 Query Editor

The query editor module is a sort of wizard tool, capable of suggesting/listing the possible values to fill a suitable query form, on the basis of the relations and the concepts described in the global ontology.

Note that even if guiding the user in writing correct queries complicates the editing process, on the other side it improves the overall performances, actually making redundant the implementation of a run-time correctness checking mechanism.

A sketch of this '*form*' is depicted in fig.2.3.

It should be pointed out that the user is left free to set many parameters and to



no results found.

Figure 2.3: Query Editor Interface

select/exclude sources from the research.

2.4.2 Presentation of Results

Once submitted a query, the user is presented the set of results as a (ranked) list of links, with indication of the name and local identifiers of the correspondent entries in the sources, and may either select one or all the entries to be entirely visualized.

The user may also save the results of the query (so to quicken future searches) and add annotations and comments to the single entry and even to single records (attributes) of the entries.

We are now working on an extension of the presentation interface, which will suggest possibly relevant queries to the user on the basis of the presented (intermediate) results.

2.5 Mediation Layer

The mediator is the core module of the system, and is responsible of parsing and decomposing (and translating) the query submitted by the user in a (ordered) sequence of source-dependent queries, then of passing these queries to the proper wrappers for execution.

2.5.1 Query Parsing and Translation

Being the needed correctness checking mechanism implicitly implemented at the query editor level, during the parsing phase the mediator has the sole task of individuating the different '*main*' concepts involved in the query (those like *'Protein'*, or *'Gene'*, which are represented by whole entries of one or more of the integrated sources).

This task is easily performed by '*reading*' the global ontology (each concept in the ontology is manually *annotated* with a *'main'* mark when appropriate), and has the purpose of separating the query into a set of sub-queries (one for each of these concepts, i.e. one for each relevant source).

The mediator then individuate the execution order, in such a way to first query the sources whose entries are needed as input for subsequent sub-queries, while making the execution parallel for those sub-queries that don't need to process intermediate results.

Sub-queries are stored as simple textual files in a predefined folder, and the file names are then passed in the established order to the wrappers for execution.

2.5.2 Processing Intermediate Results

The processing of intermediate results consists in the mediator receiving and elaborating the results of the sub-queries the wrappers performed on the remote sources, in order to use these results as input for successive sub-queries.

More precisely, the mediator captures from the entries retrieved for a sub-query the attribute values that are needed as input parameters of other sub-queries, and write them in the files representative of the sub-queries calling for these parameters.

For example, let S be a query of the kind: "select all the proteins coded by the human genes containing the sub-sequence SS", where SS is a valid string of nucleotides; let S_1, S_2 be the sub-queries "select all the genes containing the sub-sequence SS that belongs to the human species" and "select all the proteins coded by the genes in $R(S_1)$" (where $R(S_1)$ is the result set of S_1), respectively; note that S_2 requires as input parameters the results of S_1).

The mediator will submit to the appropriate wrapper/s the query S_1, then will generate a sub-query file S_2^i for each result entry R_i in $R(S_1)$ (simply using the gene name attribute in R_i), and will pass it to the proper wrapper/s.

2.6 Wrapper Layer

Wrappers are the simplest modules of the system, and are only responsible of *knowing* the actual querying mechanisms of the correspondent sources, so to exploit them to execute the (sub-)queries assigned by the mediator.

Each wrapper is passed by the mediator the path where the (sub-)query/ies is/are stored as simple textual file/s, accedes the file/s and submits the query/ies to the input modules of the correspondent source.

When the result set is returned, the wrapper generates a list of links to the retrieved entries and save it as a textual file (or generates a separate file for each entry, if they are required by the mediator as intermediate results), that can be then accessed by the mediator for presentation or further querying.

When applicable, the wrappers will exploit the indexes maintained at the source level (i.e. updated) to perform their tasks.

As an example, consider the case of the source PDB (Protein Data Bank); the source describes the structures of proteins (and other biological molecules - nucleic acids, protein-nucleic acid complexes, etc.) and their relationships to sequence, function, disease. PDB maintains indexes of the data (called summaries on the website) based on authors, resolution and components, insertion date, etc.

Exploiting these information greatly simplifies querying, when the indexed attributes are involved in the query. In fact, the wrapper can initially check for those entries that satisfy the conditions on the indexed attributes, by simply acceding the suitable indexes (publicly available on the RCSB-PDB web site [37], [36]), thus reducing the set of entries to parse. Then it can refine the research locally, by parsing only the relevant entries, looking for the satisfaction of the conditions on the remaining attributes.

2.7 Query Optimization

Being most of the queries performed by means of the tools provided by the sources themselves, while only parts of them are executed locally, talking of *'query optimization'* is not completely exact.

More often than not, optimizing a query in our system means no more than establishing the execution order of the sub-queries the query is divided into.

Nevertheless, in some cases, the bigger parts of the query has to be performed locally; this is the case of the source PDB, for example, that does not provide a proprietary querying tool, but simply a bunch of indexes, that allow the user to simplify the retrieval of entries with particular values for one or more specific attributes. Then the search has to be *refined* locally, verifying the select conditions on the other attributes involved in the query.

In these cases, optimization actually implies finding the most efficient way to answer the query (more precisely, the *refining part* of it). To this aim, some heuristics are used, that are inferred automatically during each query answering process. In practice, the system tries to measure the intrinsic *'discriminative power'* of every attributes that are characteristic of each source, in terms of the number of entries retrieved when a condition on each attribute is required, normalized on the total number of entries in the starting set.

For example, the attribute ID is certainly the most discriminative attribute possible. In fact, when asking for a particular id, only one entry will be retrieved, independently from the size of the searching space (the starting set of entries we were talking above).

Thanks to this measure of discriminative power, the system is able to decide the optimal execution order, first satisfying the conditions on the *'most discriminative'* attributes, so to greatly reduce the set of entries to analyze, then looking for the satisfaction of other, less restrictive, select conditions.

We are now working on some improvements of the query optimization, which could take into account source statistics (as average time latency, attribute's discriminative power, etc., even for sources that are completely remotely queried). These statistics are clearly going to be collected through the wrappers, being strongly related to the specific sources, and that needs a much more complicate design of these modules.

2.8 Query Processing Algorithm

In this section, we will present the formal query processing algorithm we designed. For a simple example of application, you may refer to sec. 2.1, while for a simple example of query answering in LAV, you may refer to sec. 1.3.4.

Note that the designed algorithm follow the general LAV approach of query rewriting, but under the hypothesis of inconsistent sources we chose to get the user the totality of the results, indicating for each of them the provenance, and an estimation of reliability (see sec. 2.1).

Algorithm **Query Processing**

INPUTs:
$\mathcal{Q}$ is the user query

OUTPUTs:
$\mathcal{RES}$ is the list of results, each of them being a record like: $(file\_name, results, hyperlink)$, where $file\_name$ is the name that identifies the entry containing the result data, $results$ represents the set of data that answer the user query $\mathcal{Q}$, $hyperlink$ is a link to the specific source, from which the result data are retrieved

VARs:
$SList$ is the list of relevant sources (either the ones chosen by the user, either those chosen by the mediator)
$\mathcal{S}_i$ is the i-th source in $SList$
$\mathcal{SQ}$ is the array of sub-queries $\mathcal{Q}$ is divided into
$\mathcal{SQ}_i$ is the i-th sub-query in $\mathcal{AS}$ (each sub-query specifies the source it is specifically written for)

FUNCTIONs and PROCEDUREs:
$Parse\_Query$ finds out which sources are relevant to answer $\mathcal{Q}$ (based on $SList$ - here an input/output parameter - and the mapping assertions for the queried concepts/relations)
$Divide\_Query$ gets $\mathcal{Q}$ and $SList$ as input and returns an array of sub-queries (stored in $\mathcal{SQ}$)
$Order\_SQueries$ orders the array $\mathcal{SQ}$ (based on an analysis of the existing dependencies between the sub-queries)
Pop_SQuery gets the first element of $\mathcal{SQ}$, and remove it from the array
$Call\_Wrapper$ calls for the execution of a specific wrapper (the name of the corresponding source $\mathcal{S}_i$ is an input parameter, together with the appropriate sub-query $\mathcal{SQ}_i$)
$Collect\_Results$ accedes to the result entries provided by the wrappers and creates the list of results stored in $\mathcal{RES}$
$Get\_Inconsistencies$ parses the results provided by the different wrappers, and highlights the possible inconsistencies adding an annotation to the incriminated entries in $\mathcal{RES}$

```
begin

    $SList := Parse\_Query(\mathcal{Q}, SList)$

        $\mathcal{SQ} := Divide\_Query(\mathcal{Q}, SList)$
    if $\mathcal{SQ}$ is not null
        $Order\_SQueries(\mathcal{SQ})$
        while $\mathcal{SQ}$ is not null do
            $Call\_Wrapper(Pop\_SQuery(\mathcal{SQ}))$
        end while
        $\mathcal{RES} := Collect\_Results()$            if $\mathcal{RES}$ is not null
            $\mathcal{RES} := Gets\_Inconsistencies(\mathcal{RES})$
        else write $`noresultsfound'$
        end if
    end if
end
```

In practice, the mediator accedes to the query submitted by the user and to the list of sources the user has explicitly chosen, then it individuates the integrated sources that are relevant to answer the query (when not directly specified by the user, the task is performed via the analysis of the stored mapping assertions).

Then the mediator divides the query into an *ordered* set of sub-queries (one for each relevant source), basically constituted by lists of pair $(AttributeName, Condition)$, where each attribute name is specified in the source-specific language, and the condition is in turn a pair of the kind $(Operator, Value)$ (for example, a condition on the *insertion date* of a given *protein sequence* could be $(before,$ *01-Oct-07*$)$).

Finally, the mediator calls the appropriate wrappers, and recollect the answers in a single list, that is eventually parsed to find the possible inconsistencies.

Note that from the analysis of the result data, the mediator is capable of inferring an estimation of the degree of reliability of every single result, by simply counting the number of identical results, and supposing the frequency of wrong results is always inferior to the frequency of the correct ones.

2.9 Mapping Generation Algorithm

As previously stated, the generation of the mappings between the global and the source schema is one of the most important tasks to be performed for a mediator-based integration system, being all the querying capabilities of such a system strongly dependent on how well these mappings represent the existing relationships between the attributes (concepts, in our case) in the global schema and the attributes in the source schema.

It should be noted indeed that the mappings are the basis to build suitable wrappers for the sources, and they also represent the knowledge base needed for the mediator to correctly perform its query translation task.

At the same time this task is surely the most boring and time-consuming one, it you have to perform it manually, requiring you to browse the sources, to learn how they are structured, how data are represented, and most importantly to understand the meaning of the often not even human-readable attribute names.

One of the most challenging research issues in the domain of data integration is the complete automatization of such a heavy task (something that is clearly high desirable), and works in the literature try to approach it by applying techniques from the natural language processing domain.

Almost always the existing approaches rely only on the information derivable from the schema (i.e. they don't exploit the information provided by the instances), and limit the application of the algorithms to relational and XML schema [41], [44], [54], [64]. For a good and complete classification of schema matching techniques and a comparison of some relevant methodologies, see [57] and [63].

Note that the difficulty of automating the generation of mappings is strongly dependent on how the sources are structured and on how close are the vocabularies used in the global and the source schema.

Even if we recognize the importance of automatizing this task, it is our belief that for *life sciences*, like molecular biology, where data being searched are involved in delicate researches (from drug discovery to the study of human genetic illnesses), precision and correctness are to be considered the real *'goal'*, compared to which performance considerations are probably to be deserved little attention.

With this in mind, we tried to automate as much as possible the generation of the mapping assertions, and we ended up with a (semi-)automatic procedure, where the user is asked to guide the system and validate the assertions produced.

Here is the generation algorithm [93]:

```
Algorithm **Schema Mapping Generation**

INPUTs:
$MOnto$ is the model ontology
$\mathcal{MS}$ is the stored set of sources that have been already mapped

OUTPUTs:
$\mathcal{MAP}$ is a list of arrays, each of them being the (bi-dimensional) array of mappings for a specific source

VARs:
$\mathcal{AS}$ is the source under analysis
$\mathcal{AS}_i$ is an instance (single entry) of $\mathcal{AS}$
$AttList$ is the list of all the attributes (meta-data) describing the $\mathcal{AS}$ schema
$XRef$ is the set of cross-references in an entry of $\mathcal{AS}$
$IRef$ is the set of sources in $XRef$ that have been already integrated
$\mathcal{IS}$ indicates an already integrated source
$\mathcal{IS}_j$ is an instance of $\mathcal{IS}$
```

FUNCTIONs and PROCEDUREs:
Get_Next_Source gets the next source from a given set of sources
Get_Next_Entry gets the next entry from a given source (based on the id attribute)
$Parse\_Entry$ gets the set of all the attributes (meta-data) in the source schema, given an entry from that source
$Get\_References$ gets the set of all the references in the entry being analyzed
$Get\_Referenced\_Sources$ gets the set of all those sources that have been already mapped, and are listed in $XRef$
Get_Ref_Entry gets from a referenced source the entry being referenced
Get_Common_MD gets the meta-data (attributes) that are in common between $\mathcal{IS}_j$ and $\mathcal{AS}_i$, and maps those of $\mathcal{AS}_i$ to concepts of $MOnto$
Gen_Map tries to find suitable mappings between $\mathcal{AS}$ and the concepts in $MOnto$ by applying lexical and semantic similarity functions, with the help of the user

```
begin
    if $\mathcal{MAP}$ is not empty
        $int\ counter := 0;$
        do
            $counter++;$
            $XRef := NULL$
            $\mathcal{AS}_i := Get\_Next\_Entry(\mathcal{AS})$
            if $\mathcal{AS}_i$ is not null
                $AttList := Parse\_Entry(\mathcal{AS}_i)$
                $XRef := Get\_References(\mathcal{AS}_i)$
            end if
            if $XRef$ is not empty
                $IRef := Get\_Referenced\_Sources(XRef)$
                while $(\mathcal{IS} := Get\_Next\_Source(IRef))$ is not null do
                    $\mathcal{IS}_j := Get\_Ref\_Entry(\mathcal{IS}))$
                    $\mathcal{MAP}.add(Get\_Common\_MD(\mathcal{AS}_i, \mathcal{IS}_j))$
                end while
            end if
        while $XRef$ is empty and $\mathcal{AS}_i$ is not null and $counter <= 20$
        $\mathcal{MAP}.add(Gen\_Map(\mathcal{AS}))$
    else $\mathcal{MAP}.add(Gen\_Map(\mathcal{AS}))$
    end if end
```

In words, the algorithm basically analyzes some instances of the source being mapped, and compares their attribute values with the attribute values of entries they reference, that in turn belong to already mapped sources.

In this way, the system is able to *'recycle'* the mappings obtained for other sources, thus saving the time that is needed to apply similarity-based matching algorithms, and minimizing the user intervention.

Note that in absence of already mapped sources or cross-references in the source under analysis, the application of this kind of algorithms is the only possible way of automatizing the generation process. Unfortunately, similarity-based algorithms are not always applicable with success, in particular when attribute names are not easy interpretable, and a strong user intervention is often needed to guarantee the correctness of the results.

Even if the final system should already be provided with the mappings for several sources, situations in which no source is mapped could always arise, being the user left totally free to choose which sources have to be removed/added.

If this the case, it is up to the user to select the order in which new sources are to be mapped, so to minimize the interactive steps during the whole process. Clearly it is desirable that sources with understandable attribute names and/or with a great number of external references (so to ease the mapping of subsequent sources) are the first to be mapped, so to increase the chances of simplifying future applications of the algorithm.

It should be pointed out that the generation of mapping assertions is not to be confused with the creation of wrappers for the sources, something that at this point in the work is still a task to be manually accomplished by a software engineer.

Anyway, it is possible to generate the mapping assertions before the creation of a suitable wrapper, so that the software engineer could simplify her/his work (in particular, the understanding of the source schema) with the help of the already stored mappings.

The automatic generation of wrappers is a hot research topic in the data integration field, and we plan to apply to our system the techniques and results presented in the literature in the next future.

"While working on a problem,
I never think about beauty.
I think only how to solve the problem.
But when I have finished,
If the solution is not beautiful,
I know it is wrong."
R. Buckminster Fuller.

Chapter 3

Implementation

In this chapter, we present the actual state of implementation of the system, detailing the description of what already is and what still needs to be done.

Finally, some technological notes are provided, e.g. the kind of computer used to perform the work, and the actual speed of the network connection toward the Internet.

3.1 User Interface

As we said in the previous chapter (sec. 2.4), the user interface is constituted of two modules: the *query editor*, and the *result browsing* modules.

The first *query editor* we developed was nothing more than a blank window by means of which the user could write and save any sort of query. To the aim of quickening the execution phase, queries were not checked for correctness, and this could sometimes generate problems to the less experienced user.

To solve this problem, we worked on a new version that basically consists of a form to be compiled, each of the possible fields being checked onto the model ontology to ensure the correctness of the resulting query.

At the present state of implementation the query editor is a sort of wizard tool (see fig.2.3). The user browses the global ontology to identify the main concepts she/he wishes to query (i.e. the set of relevant sources), then the system cuts the relations and concepts that are not related to the chosen concepts and presents the user a limited portion of the ontology (the portion around the main concepts chosen) for further selection of concepts/relations, thus in practice forcing the user to submit only meaningful queries.

Once the query is written, the system asks the user if she/he prefers to exclude any of the possibly relevant sources, and if she/he wants to specify which sources have to be given high reliability in case of inconsistencies.

The current work on the user interface mainly involves the *result browsing* module. The interface now simply presents to the user a list of (ranked) results, with indication of the correspondent sources. The ultimate version will give the chance of further querying the set of results presented, and to rearrange the presentation of the results on the basis of different parameters (source alphabetical order, last modification date of the entry, etc.)

3.2 Mediator Module

The implementation of the mediator module has been the core of the whole implementation work, and proceeded in parallel with the implementation of the two wrapper modules currently part of the system.

At the present state of the work, the mediator consists of several software modules, written in Java, each of which performs one of the task the mediator has to accomplish. Note that these modules are characterized by a large number of classes and methods, and in turn use objects from other suitably defined classes to perform their tasks.

Furthermore most of the mediation tasks are not performed by single software modules, but require the instantiation of objects from different classes, so talking of *'module performing a task'* is not always strictly correct. Broadly speaking (in the sense that some of them are rather collection of classes), we can identify the following central *'modules'*:

- *Translator* - responsible of parsing submitted queries, rewriting them in a set of source-dependent sub-queries, and calling the suitable wrappers for execution; the main classes that constitute the Translator are:

  - queryParser, which accedes the user query (stored after editing as a list of conditions on concepts of the global schema), tokenizes it and stores it in a suitable array (each element of the array being a record of the kind $(concept, condition)$);
  - queryRewriter, which is passed the array representing the query, and performs the tasks of rearranging the records into a set of arrays (each corresponding to a different source needed to answer the conditions), and of rewriting the records in a shape like $(attribute, condition)$, where now the conditions are expressed on source's attributes, rather than on concepts of the global ontology.

- *Optimizer* - at the time of writing its sole purpose is to suitably arrange the execution order of the sub-queries, so to timely retrieve intermediate results, before calling the wrappers that needs them for their task.
- *Checker* - responsible of checking for inconsistencies among the retrieved results; it basically looks at the results and highlights those that are more probably erroneous (on the basis of statistical reasoning and of the indications provided by the user at query editing time - see sec. 3.1).

Most of the current implementation work involves the Optimizer and the Checker modules, to the aim of: enriching the former with the capability of inferring source statistics (toward an effective query optimization) and measuring the *discriminative* power of the most queried ontology concepts; providing the latter with more powerful means to measure the reliability of a given source (something that actually concerns design more than implementation).

3.3 Wrapper Modules

The system now has two wrappers implemented, corresponding to two important sources of protein data (as we said in sec. 2.3.1, we limited the development of the model ontology to a contour of the *Protein* concept), UniProtKB/Swiss-Prot [?] for sequences, and RCSB PDB (Protein Data bank) [37] for structures (actually more than this [36]).

3.3.1 UniProtKB/Swiss-Prot

For the development of the wrapper Swiss-Prot, we largely relied on the functionality offered by the SRS wrapping system. In practice SRS (Sequence retrieval System) [52] maintains some updated indexes on most of the Swiss-Prot schema attributes, and provide for a world wide web interface, by means of a program called WGETZ (it is basically a CGI-script) [53], that allows to accede to the indexed entries by simply opening HTML links.

On the basis of these information, the Swiss-Prot wrapper performs its task in three main steps, as described below.

In the first step, the wrapper divides the sub-query the mediator assigned to it in two distinct parts: one involving attributes that are indexed by SRS (i.e. retrievable through WGETZ), the other involving attributes that are not indexed, and need the actual parsing of the single entries to be answered.

In the second step, the wrapper creates an HTML link to WGETZ which would answer the first part of the sub-query, open it and stores the results as textual files in a established path.

In the third and last step, the wrapper parses the retrieved entries and look for the satisfaction of the conditions expressed in the second part of the sub-query, deleting the entries that do not satisfy them, and keeping the others.

3.3.2 RCSB PDB

The wrapper for the PDB works in a very similar way to that described above, in that it exploits the indexing facilities provided by the PDB on many of the attributes describing the entries.

The case here is complicated by the absence of a remote access tool like WGETZ. Hence, the wrapper (after dividing the sub-query in the same way as the Swiss-Prot wrapper does) must accede to the indexes via FTP and browse them locally, then it must directly retrieve all the relevant entries, and save them locally for the final parsing.

3.4 Technological Notes

The computer used during development and experiments is characterized by the following configuration: CPU Intel P4 2.99 GHz; DDR RAM PC3200 2048 MB; OS MS Windows XP Professional Edition SP2.

The network adapter installed on the machine is a Broadcom NetXtreme Gigabit Ethernet; the real (i.e. tested) average upload/download network speeds to/from Internet are approximately 12/14 (with peaks of 13/15) Mbit/sec, so we can confidently say that the experiments we performed were barely influenced by our network performances, while certainly were more dependent from the sources' response times.

"Experience is a hard teacher because she
Gives the test first, the lesson afterwards."

Vernon Sanders Law.

Chapter 4

Experimental Work and Results

In this chapter we will discuss the results of the experiments we performed in order to prove the effectiveness of our approach to the problem of biological data integration, in particular to the problem of querying overlapping sources.

As a side note, while experimenting the system most of the automatic OS tool services (print spooling, automatic configuration services, firewall, etc.) and most of the utilities normally running on the PC (e.g. anti-virus software, performances monitor, etc.) were disabled.

4.1 Query Processing

As for query processing, we tested the effectiveness of the system in retrieving the results from the two wrapped sources, when the submitted user query requires data from both them.

Note that testing the performances of the system would have resulted in a meaningless set of temporal values, given that to the best of our knowledge in the literature no such results are provided, so we would have lacked the reference benchmark.

As a side note, we observed anyway that the *absolute* delay in retrieving the results to the submitted queries were acceptable, but greatly dependent on the network performances (the same set of queries, performed on different days - but close in time, so that was presumably not dependent on variation of the sources' content -, required different - sometimes conspicuously - amounts of time).

4.1.1 Experiment Details

Since in the literature there exists no benchmark to compare the experimental results with, for testing the effectiveness of the system we needed to create a kind of benchmark on our own.

Given that the system has the sole purpose of facilitating the research tasks biologists need to perform on the Web, we decided to compare the results of the execution of ten different queries with those obtained by hand by a group of computer engineers with proved practice of web surfing.

Note that we did not enroll a group of experienced biologists for two important reasons: first, we wanted to see if the system was able to retrieve more (correct) results than a person with a poor knowledge of the representational formats of the two integrated sources, and enrolling domain experts would have probably meant they already knew Swiss-Prot and PDB; furthermore, we wanted to have some hints about the capability of the system of simplifying the retrieval process, and to this purpose we needed the opinion of people with a good experience in information retrieval.

4.1.2 Discussion

The results of the experiments showed that the system succeded in retrieving the totality of data which correctly answered the user queries. Feedback from the users involved in the experiments confirmed that the use of the system proved effective both for simplifying the whole retrieval process, and for the interpretation of the results obtained (this is certainly dependent from their poor knowledge of the representational formats of the sources - in particular of the barely readable Swiss-Prot two-letter attribute codes).

A comparative test with domain experts will probably be useful when the number of integrated sources will increase to values greater than a dozen, whereas even domain experts will probably find it difficult to understand every source's format and to resolve every semantic eterogeneity (especially when the sources will not be cross referencing one each other).

As an important note, it should be pointed out that the system averagely took less time to answer the queries than the students took working by hand, in particular when the queries involved non-indexed attributes - i.e. requiring the students to manually parse the result set (obtained by querying the sources on indexed attributes) to check for the satisfaction of the query conditions on these attributes.

"How to make God laugh:
Tell him your future plans."

Woody Allen.

Chapter 5

Conclusions and Future Works

In this thesis we presented an ontology-based integration system for biological data sources. We analyzed the issue and the possible methodologies for tackling it, presented the state-of-the-art and finally showed how our system is capable of answering the integration needs of molecular biology researchers.

The system is still under implementation, much work is still needed in different directions, and is going to be object of future developments:

- *The Global Ontology needs to be completed and validated with common efforts of software engineers and domain experts.*
- *The User Interface can be improved with respect to the presentation of the query results.*
- *The Mediator can be enriched with suitable means for collecting sources' statistics, so to improve the query optimization task, and for automatically validating the data retrieved from the sources, so to avoid the presentation of inconsistent (and incorrect) results.*
- *Many other sources still are to be integrated to provide a satisfiable level of effectiveness in answering user queries, and suitable wrappers and mappings need to be provided for them.*

Much experimental work also needs to be performed, in order to completely validate the approach we followed. In particular, work is still to be dedicated to a formal measurement of the global performances of the system, a task that we plan to accomplish when the number of integrated source will increase to big numbers, and that we hope will prove useful to the community as initial benchmarking attempt in the literature.

It should be also pointed out that the greater the number of integrated sources the higher the time needed for rewriting user queries and reconciling data after retrieval. Hence, a study of the scalability of the approach needs to be performed, something that also is missing in the literature, where the (mediator-based) integration system presented limit their coverage to a very small number of sources (usually inferior to ten).

Furthermore, future work may still be devoted to the development of a means for automatically producing wrappers, and the results of such a research may lead to the partial redesign of important parts of the mediator (e.g., the way queries are rewritten, the way schema mappings are generated, etc.).

Finally, some work should be dedicated to make the system work in multithreading, parallelizing as much as possible the execution of the sub-queries at the source level.

Something that became more and more clear while working on the system design and development is that the common efforts of several (and possibly different, from a cultural background point of view) minds work always better than a single-minded researcher can, for it is impossible for a single person to manage all the possible aspects of a complicate multidisciplinary problem, and more often than not the help of somebody looking from a different perspective is useful to recognize the limits and *defects of the particular, hence to improve/correct the whole.*

"Whenever people agree with me I always feel I must be wrong."

Oscar Wilde.

Appendix A

Resource Description Framework and its Extensions

This Appendix is designed to provide the reader with the basic knowledge required to effectively use RDF. It introduces the basic concepts of RDF and describes its XML syntax. Furthermore, it provides an analysis of some of the most relevant standard extensions presented in the literature. Most of introductory section references [82].

A.1 W3C RDF Standard

The Resource Description Framework (RDF) is a language for representing information about *resources* in the World Wide Web. It is particularly intended for representing metadata about Web resources, such as the title, author, and modification date of a Web page, copyright and licensing information about a Web document, or the availability schedule for some shared resource.

However, by generalizing the concept of a *Web resource*, RDF can also be used to represent information about things that can be identified on the Web, even when they cannot be directly retrieved on the Web. Examples include information about items available from on-line shopping facilities (e.g., information about specifications, prices, and availability), or the description of a Web user's preferences for information delivery.

RDF is intended for situations in which this information needs to be processed by applications, rather than being only displayed to people. RDF provides a common framework for expressing this information so it can be exchanged between applications without loss of meaning.

Since it is a common framework, application designers can leverage the availability of common RDF parsers and processing tools. The ability to exchange information between different applications means that the information may be made available to applications other than those for which it was originally created.

RDF is based on the idea of identifying things using Web identifiers (called Uniform Resource Identifiers, or URIs), and describing resources in terms of simple properties and property values. This enables RDF to represent simple statements about resources as a graph of nodes and arcs representing the resources, and their properties and values. To make this discussion somewhat more concrete as soon as possible, the group of statements "there is a *Person* identified by $http://www.w3.org/People/EM/contact\#me$, whose name is *Eric Miller*, whose email address is $em@w3.org$, and whose title is *Dr.*" could be represented as the RDF graph in figure A.1.

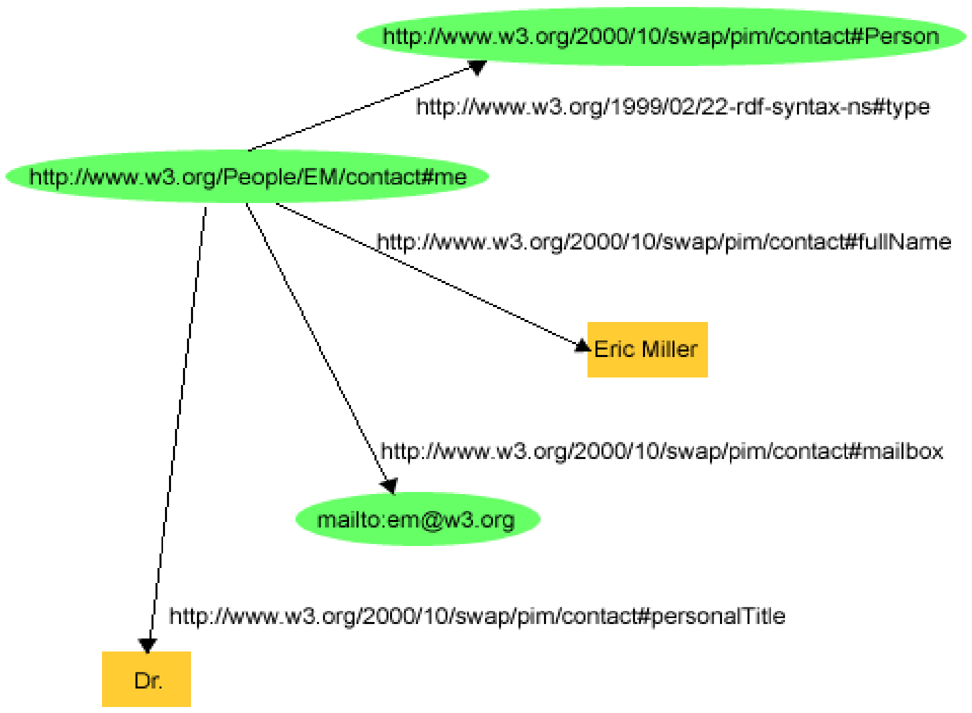

http://www.w3.org/2000/10/swap/pim/contact#Person
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
http://www.w3.org/People/EM/contact#me
http://www.w3.org/2000/10/swap/pim/contact#fullName
Eric Miller
http://www.w3.org/2000/10/swap/pim/contact#mailbox
mailto:em@w3.org
http://www.w3.org/2000/10/swap/pim/contact#personalTitle
Dr.

Figure A.1: Example of RDF graph, describing Eric Miller.

The figure illustrates that RDF uses URIs to identify:

- individuals, e.g. *Eric Miller*, identified by $http://www.w3.org/People/EM/contact\#me$
- kinds of things, e.g. *Person*, identified by $http://www.w3.org/2000/10/swap/pim/contact\#Person$
- properties of those things, e.g. *mailbox*, identified by $http://www.w3.org/2000/10/swap/pim/contact\#mailbox$
- values of those properties, e.g. $mailto : em@w3.org$ as the value of the *mailbox* property (RDF also uses character strings such as 'Eric Miller', and values from other datatypes such as integers and dates, as the values of properties)

RDF also provides an XML-based syntax (called RDF/XML) for recording and exchanging these graphs. Here is a small chunk of RDF in RDF/XML corresponding to the graph in the figure above:

```
<? xml version= "1.0"? >
<rdf:RDF xmlns:rdf= "http : //www.w3.org/1999/02/22 − rdf − syntax − ns#"
        xmlns:contact= "http : //www.w3.org/2000/10/swap/pim/contact#" >

    <contact:Person rdf:about= "http : //www.w3.org/People/EM/contact#me" >
        <contact:fullName>Eric Miller< /contact:fullName>
        <contact:mailbox rdf:resource= "mailto : em@w3.org"/ >
        <contact:personalTitle>Dr.< /contact:personalTitle>
    < /contact:Person>

< /rdf:RDF>
```

Like HTML, this RDF/XML is machine processable and, using URIs, can link pieces of information across the Web. However, unlike conventional hypertext, RDF URIs can refer to any identifiable thing, including things that may not be directly retrievable on the Web (such as the person Eric Miller).

As stated above, the result is that in addition to describing such things as Web pages, RDF can also describe cars, businesses, people, news events, etc. (i.e. *real things*). In addition, RDF properties themselves have URIs, to precisely identify the relationships that exist between the linked items.

For a detailed specification of the RDF/XML Syntax, RDF Semantics, and RDF

Vocabulary Description Language (RDF Schema), refers to [83], [84], and [85], respectively.

A.2 RDF Extensions

A plethora of extensions to the RDF standard have been proposed since the W3C specifications came out in the late 2004, that try to solve the representational issues that is not possible to tackle with the simple standard syntax or vocabulary.

The most relevant extensions, with respect to the practical use of RDF in representing web resources, are certainly those that try to address the problem of representing the temporal dimension.

A.2.1 Temporal RDF (t-RDF)

In the RDF model, the universe to be modeled is a set of resources, essentially anything that can have a universal resource identifier, URI. The language to describe them is a set of properties, technically binary predicates. Descriptions are statements very much in the subject-predicate-object structure.

Although some studies exist about addressing changes in an ontology, or the need for temporal annotations on Web documents, little attention has deserved the problem of representing, updating and querying temporal information in RDF.

But, as pointed out by Abiteboul [21], the modeling of time is one of the key primitives needed in a query language for Web and semistructured data. On this basis, the application of temporal database concepts to RDF to allow metadata navigation across time led to the development of several extensions of the standard, fundamentally based on *time labeling* or *versioning* approaches - the former consists in labeling the elements subject to changes (i.e. triples), the latter is based on maintaining a snapshot of each state of the graph.

Note that there are at least two temporal dimensions to consider when dealing with temporal databases: *valid* and *transaction* times. Valid time is the time when data is valid in the modeled world; transaction time is the time when data is actually stored in the database. The versioning approach captures transaction time, while labeling is mostly used when representing valid time.

A good example of such an extension is provided by Gutierrez [88], whose approach supports both time dimensions.

Bibliography

[1] M. R. Quillian: *'Word concepts: A theory and simulation of some basic semantic capabilities'*, Behavioral Science, vol. 12, pp. 410-430, 1967.

[2] M. Minsky: *'A framework for representing knowledge'*, The Psychology of Computer Vision, McGraw-Hill, New York, 1975.

[3] D. Bobrow, and T. Winograd: *'An overview of KRL, a knowledge representation language'*, Cognitive Science, vol. 1, is. 1, 1977.

[4] M. R. Garey, and D. S. Johnson: *'Computers and Intractability: A Guide to the Theory of NP-Completeness'*, W.H. Freeman and Co., San Francisco, CA, 1979.

[5] R. Brachman, and J. Schmolze: *'An overview of the KL-ONE knowledge representation system'*, Cognitive Science, vol. 9, is. 2, 1985.

[6] M. Genesereth, and N. Nilsson: *'Logical Foundations of Artificial Intelligence'*, Morgan Kaufmann, San Mateo, CA, 1987.

[7] J. W. Lloyd: *'Foundations of Logic Programming'*, Springer-Verlag, Berlin, 1987.

[8] D. Lenat, R. V. Guha, K. Pittman, D. Pratt, and M. Shepherd: *'Cyc: Toward programs with common sense'*, Communications of the ACM, vol. 33, no. 8, pp. 30-49, 1990.

[9] R. V. Guha: *'Contexts: A Formalization and Some Applications'*, Ph.D. Thesis Dissertation, Department of Computer Science, Stanford University, 1991.

[10] R. Davis, H. Shrobe, and P. Szolovits: *'What Is a Knowledge Representation?'*, AI Magazine, pp. 17-33, AAAI, 1993.

[11] T. R. Gruber: *'A translation approach to portable ontology specifications'*, Knowledge Acquisition, vol. 5, no. 2, pp. 199-220, 1993.

[12] T. R. Gruber: *'Towards principles for the design of ontologies used for knowledge sharing'*, Proceedings of the International Workshop on Formal Ontology, Padova, Italy, 1993. *Available as technical report KSL-93-04, Knowledge Systems Laboratory, Stanford University.* $ftp.ksl.ftanford.edu/pub/KSL_Reports/KSL - 983 - 04.ps$.

[13] J. McCarthy: *'Notes on formalizing context'*, Proceedings of the 13th International Conference on Artificial Intelligence (IJCAI-93), pp. 555-560, Los Altos, Morgan Kaufmann, 1993.

[14] A. Gomez-Perez: *'Some ideas and examples to evaluate ontologies'*, Technical Report KLS-94-65, Knowledge Systems Laboratory, Stanford, 1994.

[15] S. Abiteboul, R. Hull, and V. Vianu: *'Foundations of Databases'*, Addison Wesley, Publ. Co., Reading, Massachussetts, 1995.

[16] P. Buneman, S. Davidson, K. Hart, C. Overton, and L. Wong: *'A Data Transformation System for Biological Data Sources'*, Proceedings of the 21st International Conference on Very Large Data Bases (VLDB1995), 1995.

[17] S. Davidson, C. Overton and P. Buneman: *'Challenges in Integrating Biological Data Sources'*, Journal of Computational Biology, vol. 2, no. 4, 1995.

[18] N. Guarino, and P. Giaretta: *'Ontologies and knowledge bases: Towards a terminological clarification'*, Towards Very Large Knowledge Bases: Knowledge Building and Knowledge Sharing, pp. 25-32, IOS Press, Amsterdam, 1995.

[19] P. Karp: *'A Strategy for Database Interoperation'*, Journal of Computational Biology, vol. 2, no. 4, pp. 573-586, 1995.

[20] M. Uschold, and M. Gruninger: *'Ontologies: principles, methods and applications'*, Knowledge Engineering Review, vol. 11, no. 2, pp. 93-136, 1996.

[21] S. Abiteboul: *'Querying Semi-Structured Data'*, Proceedings of the 6th International Conference on Database Theory (ICDT'97), Delphi, Greece, 1997.

[22] M. Fernandez, A. Gomez-Perez, and N. Juristo: *'METHONTOLOGY: from Ontological Art towards Ontological Engineering'*, Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, pp. 33-40, Stanford, US, Mar 1997.

[23] J. D. Ullman: *'Information integration using logical views'*, Proceedings of the 6th Int. Conf. on Database Theory (ICDT'97), Lecture Notes in Computer Science, vol. 1186, pp. 19-40, Springer-Verlag, 1997.

[24] S. Abiteboul, and O. Duschka: *'Complexity of answering queries using materialized views'*, In Proceedings of the 17th ACM SIGACT SIGMOD

SIGART Symposium on Principles of Database Systems (PODS'98), pp. 254-265, 1998.

[25] P. Baker, A. Brass, S. Bechhofer, C. Goble, N. Paton, and R. Stevens: *'TAMBIS: Transparent Access to Multiple Bioinformatics Information Sources'*, Proceedings of the 6th International Conference on Intelligent Systems for Molecular Biology (ISMB98), 1998.

[26] D. A. Duce, and G. A. Ringland: *'Approaches to Knowledge Representation: An Introduction'*, John Wiley, Chichester, 1988.

[27] N. Guarino: *'Formal ontology and information systems'*, Proceedings of Formal Ontology and Information Systems, Trento, Italy, IOS Press, Jun 1998.

[28] D. M. Jones, T. J. M. Bench-Capon, and P. R. S. Visser: *'Methodologies For Ontology Development'*, Proceedings of the IT&KNOWS Conference, XV IFIP World Computer Congress, Budapest, Aug 1998.

[29] R. B. Altman, M. Bada, X.-J. Chai, M. W. Carillo, R. Chen, and N. F. Abernethy: *'RiboWeb: An Ontology-Based System for Collaborative Molecular Biology'*, IEEE Intelligent Systems and Their Applications, vol. 14, no. 5, pp. 68-76, 1999.

[30] M. Arenas, L. E. Bertossi, and J. Chomicki: *'Consistent query answers in inconsistent databases'*, Proceedings of the 18th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'99), pp. 68-79, 1999.

[31] P. G. Baker, C. A. Goble, S. Bechhofer, N. P. Paton, R. Stevens, and A. Brass: *'An ontology for bioinformatics applications'*, Bioinformatics, vol. 15, no. 6, pp. 510-520, 1999.

[32] M. Friedman, A. Levy, and T. Millstein: *'Navigational plans for data integration'*, Proceedings of the 16th National Conference on Artificial Intelligence (AAAI'99), AAAI Press/The MIT Press, pp. 67-73, 1999.

[33] A. Ouksel and I. Ahmed: *'Ontologies Are Not the Panacea in Data Integration'*, Journal of Distributed and Parallel Databases, vol. 7, pp. 1-29, 1999.

[34] N. Paton, R. Stevens, P. Baker, C. Goble, S. Bechhofer, and A. Brass: *'Query Processing in the TAMBIS Bioinformatics Source Integration System'*, Proceedings of the 11th International Conference on Scientific and Statistical Database Management (SSDBM1999), Cleveland, Ohio, USA, pp. 138-147, IEEE, Jul 1999.

[35] Ashburner et al.: *'Gene Ontology: Tool for the Unification of Biology'*, Nature Genetics, vol. 25, pp. 25-29, 2000.

[36] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne: *'The Protein Data Bank'*, Nucleic Acids Research, vol. 28, pp. 235-242, 2000.

[37] *'RCSB Protein Data Bank'*, *http://www.rcsb.org/pdb/home/home.do*, last accessed on Nov 2007.

[38] C. A. Goble: *'Supporting Web-based Biology with Ontologies'*, Proceedings of the 3rd IEEE International Conference on Information Technology Applications in Biomedicine (ITAB'00), Arlington, VA, November 2000, pp. 384-390.

[39] *'The Gene Ontology Consortium'*, *http://www.geneontology.org/*, last accessed on Nov 2007.

[40] D. T. Jones: *'Protein structure prediction in the postgenomic era'*, Current Opinion in Structural Biology, vol. 10, pp. 371-379, Elsevier, 2000.

[41] D. Beneventano, et al.: *'Information Integration: the MOMIS Project Demonstration'*, Proceedings of the 26th International Conference on Very Large Data Bases (VLDB2000), pp. 611-614, Cairo, Egypt, 2000.

[42] R. Stevens, C. A. Goble, and S. Bechhofer: *'Ontology-based Knowledge Representation for Bioinformatics'*, Briefings in Bioinformatics, vol. 1, no. 4, pp. 398-414, 2000.

[43] D. Bader, B. M. Moret, and L. Vawter: *'Industrial applications of high-performance computing for phylogeny reconstruction'*, Proceedings of SPIE Commercial Applications for High-Performance Computing, vol. 4528, pp. 159-168, Denver, CO, Aug 2001.

[44] R. J. Miller, et al.: *'The Clio project: managing heterogeneity'*, ACM SIGMOD Record, vol. 30, is. 1, pp. 78-83, Mar 2001.

[45] S. Davidson, J. Crabtree, B. Brunk, J. Schug, V. Tannen, C. Overton and C. Stoeckert: *'K2/Kleisli and GUS: Experiments in Integrated Access to Genomic Data Sources'*, IBM Systems Journal, vol. 40, no. 2, pp. 512-531, 2001.

[46] C. A. Goble, R. Stevens, G. Ng, S. Bechhofer, N. W. Paton, P. G. Baker, M. Peim, and A. Brass: *'Transparent Access to Multiple Bioinformatics Information Sources'*, IBM Systems Journal, vol. 40, no. 2, pp. 532-551, 2001.

[47] *'The TAMBIS Project, Tutorial and Demos'*, *http://www.cs.man.ac.uk/ stevensr/tambis/text/details.html*

[48] G. Greco, S. Greco, and E. Zumpano: *'A logic programming approach to the integration, repairing and querying of inconsistent databases'*, Proceedings of the 17th International Conference on Logic Programming (ICLP'01), Lecture Notes in Artificial Intelligence, Springer, vol. 2237 , pp. 348-364, 2001.

[49] L. Haas, P. Schwarz, P. Kodali, E. Kotlar, J. Rice and W. Swope: *'DiscoveryLink: A System for Integrated Access to Life Sciences Data Sources'*, IBM Systems Journal, vol. 40, no. 2, pp. 489-511, 2001.

[50] A. Y. Halevy: *'Answering queries using views: A survey'*, The VLDB Journal, vol. 10, no. 4, pp. 270-294, 2001.

[51] J. D. Heflin: *'Towards the Semantic Web: Knowledge Representation in a Dynamic, Distributed Environment'*, Ph.D. Thesis Dissertation, Department of Computer Science, University of Maryland, College Park, 2001.

[52] R. Lopez: *'SRS - Sequence Retrieval System'*, Presentation, *http://www.pdg.cnb.uam.es/cursos/BioInfo2001/pages/SRS/index.html*, Universidad Autonoma de Madrid, last accessed on Oct 2007.

[53] *'Icarus Documentation'*, *http://www.expasy.org/srs5/man/srsman.html*, last accessed on Nov 2007.

[54] J. Madhavan, P. A. Bernstein, and E. Rahm: *'Generic Schema Matching with Cupid'*, Proceedings of the 27th International Conference on Very Large Data Bases (VLDB2001), pp. 49-58, Roma, Italy, 2001.

[55] P. Mork, A. Y. Halevy, and P. Tarczy-Hornoch: *'A Model for Data Integration Systems of Biomedical Data Applied to Online Genetic*

Databases', Proceedings of the Symposium of the American Medical Informatics Association, 2001.

[56] N. Paton, and C. Goble: *'Information Management for Genome Level Bioinformatics'*, VLDB 2001 Tutorial, 2001.

[57] E. Rahm, and P. A. Bernstein: *'A survey of approaches to automatic schema matching'*, The VLDB Journal, vol. 10, no. 4, pp. 334-350, Dec 2001.

[58] W. Sujansky: *'Heterogeneous Database Integration in Biomedecine'*, Methodological Review, Journal of Biomedical Informatics, vol. 34, pp. 285-298, 2001.

[59] *'W3C Semantic Web FAQs'*, *http://www.w3.org/2001/sw/SW-FAQ#What1*, last accessed on Nov 2007.

[60] *'W3C Semantic Web Activity'*, *http://www.w3.org/2001/sw/#spec*, last accessed on Nov 2007.

[61] L. Bertossi, J. Chomicki, A. Cortes, and C. Gutierrez: *'Consistent answers from integrated data sources'*, Proceedings of the 6th International Conference on Flexible Query Answering Systems (FQAS 2002), pp. 71-85, 2002.

[62] A. Cali, D. Calvanese, G. De Giacomo, and M. Lenzerini: *'On the Expressive Power of Data Integration Systems'*, Proceedings of the 21st International Conference on Conceptual Modeling (ER 2002), 2002.

[63] H.-H. Do, S. Melnik, and E. Rahm: *'Comparison of Schema Matching Evaluations'*, Proceedings of the 2nd International Workshop on Web Databases (German Informatics Society), 2002.

[64] K. Jagannathan: *'An Approach to Schema Mapping Generation for Data Warehousing'*, Master Thesis Dissertation, Department of Computer Science and Engineering, University of Texas, Arlington, Dec 2002.

[65] M. Lenzerini: *'Data integration: A theoretical perspective'*, Proceedings of the 21st ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS 2002), pp. 233-246, 2002.

[66] M. L. Metzker, D. P. Mindell, X.-M. Liu, R. G. Ptak, R. A. Gibbs, and D. M. Hillis: *'Molecular evidence of HIV-1 transmission in a criminal case'*, PNAS, vol. 99, no. 2, pp. 14292-14297, 2002.

[67] M. Pop, S. L. Salzberg, and M. Shumway: *'Genome Sequence Assembly: Algorithms and Issues'*, Computer, vol. 35, is. 7, pp. 47-54, IEEE, Jul 2002.

[68] A. D. Baxevanis: *'TheMolecular Biology Database Collection: 2003 Update'*, Nucleic Acids Research, vol. 31, no. 1, pp. 1-12, 2003.

[69] L. Bravo, and L. Bertossi: *'Logic programming for consistently querying data integration systems'*, Proceedings of the 18th International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 10-15, 2003.

[70] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa: *'Data exchange: Semantics and query answering'*, Proceedings of the 9th International Conference on Database Theory (ICDT 2003), pp. 207-224, 2003.

[71] M. A. Harris, and H. Parkinson: *'Standards and ontologies for functional genomics: towards unified ontologies for biology and biomedicine'*, conference report, Comparative and Functional Genomics, vol. 4, pp. 116-120, John Wiley & Sons, Ltd., 2003.

[72] Z. Lacroix, F. Naumann, L. Raschid, and M. E. Vidal, *'Exploring Life Sciences Data Sources'*, Proceedings of IJCAI-03 Workshop on Information Integration on the Web, 2003.

[73] P. Lambrix, M. Habbouche, and M. Pérez: *'Evaluation of Ontology Development Tools for Bioinformatics'*, Bioinformatics, vol. 19, no. 12, pp. 1564-1571, 2003.

[74] Z. Ben Miled, N. Li, M. Baumgartner and Y. Liu: *'A Decentralized Approach to the Integration of Life Science Web Databases'*, Informatica, vol. 27, no. 1, 2003.

[75] J. D. Watson, and A. Berry: *'DNA: The Secret of Life'*, Knopf, 2003.

[76] T. L. Wiliams, and B. M. E. Moret: *'An Investigation of Phylogenetic Likelihood Methods'*, Proceedings of the 3rd IEEE Symposium on Bioinformatics and Bioengineering (BIBE'03), pp. 79-86, Mar 2003.

[77] J. B. L. Bard, and S. Y. Rhee: *'Ontologies in Biology: Design, Applications and Future Challenges'*, Nature Reviews/Genetics, vol.5, pp. 213-222, Mar 2004.

[78] A. Calì, D. Calvanese, G. De Giacomo, and M. Lenzerini: *'Data integration under integrity constraints'*, Proceedings of the 14th Conference on Advanced Information Systems Engineering (CAiSE 2002), Information Systems, vol. 29, no. 2, pp. 147-163, Elsevier, Apr 2004.

[79] J. Cohen: *'Bioinformatics - An Introduction for Computer Scientists'*, ACM Computer Surveys, vol. 36, no. 2, pp. 122-158, Jun 2004.

[80] T. Hernandez, and S. Kambhampati: *'Integration of Biological Sources: Current Systems and Challenges Ahead'*, ACM SIGMOD Record, vol. 33, is. 3, pp. 51-60, Sep 2004.

[81] D. Lembo: *'Dealing with Inconsistencies and Incompleteness in Data Integration'*, Ph.D. Thesis Dissertation, Department of Computer Science, Università degli Studi di Roma 'La Sapienza', 2004.

[82] *'W3C Resource Description Framework Primer'*, *http://www.w3.org/TR/rdf-primer/*, Feb 10th 2004, last accessed on Nov 2007.

[83] *'W3C RDF/XML Syntax Specification'*, *http://www.w3.org/TR/rdf-syntax-grammar/*, Feb 10th 2004, last accessed on Nov 2007.

[84] *'W3C RDF Semantics'*, *http://www.w3.org/TR/2004/REC-rdf-mt-20040210/*, Feb 10th 2004, last accessed on Nov 2007.

[85] *'W3C RDF Vocabulary Description Language 1.0: RDF Schema'*, *http://www.w3.org/TR/2004/REC-rdf-schema-20040210/*, Feb 10th 2004, last accessed on Nov 2007.

[86] D. Caragea, J. Bao, J. Pathak, A. Silvescu, C. Andorf, D. Dobbs, and V. Honavar: *'Information Integration from Semantically Heterogeneous Biological Data Sources'*, Proceedings of the 16th International workshop on Database and Expert Systems Applications (DEXA'05), pp. 580-584, Aug 2005.

[87] *'The GUS Platform for Functional Genomics'*, *http://www.gusdb.org*, last accessed on Oct 2007.

[88] C. Gutierrez, C. Hurtado, and A. Vaisman: *'Temporal RDF'*, Proceedings of the 2nd European Semantic Web Conference (ESWC2005), Lecture Notes in Computer Science, vol. 3532, pp. 93-107, Springer Berlin, 2005.

[89] V. Jakoniene, and P. Lambrix: *'Ontology-based integration for bioinformatics'*, Proceeding of the 31st International Conference on Very Large Data Bases (VLDB2005), Trondheim, Norway, 2005.

[90] Z. Kedad, and X. Xue: *'Mapping generation for XML data sources: a general framework'*, Proceedings of the 2005 International Workshop on Challenges in Web Information Retrieval and Integration (WIRI'05), IEEE, 2005.

[91] J. Moult: *'A decade of CASP: progress, bottlenecks and prognosis in protein structure prediction'*, Current Opinion in Structural Biology, vol. 15, pp. 285-289, Elsevier, 2005.

[92] J. Tauberer: *'What Is RDF'*, on O'Reilly's xml.com, *http://www.xml.com/pub/a/2001/01/24/rdf.html?page=1*, Jul 26th 2006, last accessed on Nov 2007.

[93] P. Capasso, and A. Picariello: *'BioInView: Integration and Querying of Heterogeneous Biological Data Sources'*, Proceedings of the 2007 International Conference on Bioinformatics & Computational Biology (BIOCOMP'07), *in press*, Las Vegas, Nevada, US, Jun 2007.

[94] *'EMBL Nucleotide Sequence Database Statistics'*, *http://www3.ebi.ac.uk/Services/DBStats*, last accessed on Nov 2007.

[95] *'Ontology in Computer Science'*, *http://en.wikipedia.org/wiki/Ontology_(computer_science)*, last accessed on Nov 2007.

Special Thanks

To proff. Antonio Picariello, Angelo Chianese, Lucio Sansone.. for continuously stimulating my mind and instilling doubts and ideas that helped me to grow as a man and as a scholar; for being inspiration and example; for helping me to face any challenges with no discouragement.

To prof. Luigi Pietro Cordella.. for being always patient and ready to answer any question and solve any problem for me and my colleagues.

To prof. VS Subrahmanian.. for giving me the opportunity of working in a wonderful place, with wonderful people.

To Francesca.. simply for being there.. always.

To Rosi, Sasà, Cristina.. for patiently and fearlessly listening to my complaints on pretty much everything; for having shared with me all the lunch breaks with a harm smile ready; for the wonderful nights at the cinema; for our dinners at the 'Babette'.

To Antonio Maria, Antonio, Vinni, Carmine, and the newbies Andrea, Francesco, Luigi, and Sergio (great colleagues and friends).. for teaching me what the term 'hard worker' means; for the football matches; for the good time spent together.

To Amy, Maria Esther, Diego, Octavian.. for their help and their sincere encouragement during those great days.

To Ciccio and Dodò.. my closest friends.. for helping me understand the distance is not a matter.

To Lorenzo, Vincenzo, Luca and Olga, Fabreeze, il Morbido Giallone, il Campadrino, Ben Grimm, il Testone, Peppone 'e Nola, Nespà, o'Cinese, and all the other magic/poker guys i'm forgetting now (be pleasant, you know i'm burned).. for making me always enjoy the time with you; i hope it is the same for you.

To my youngest brother Brandon, and to his family.. for being so close, even if so far.. with you i felt like i was at home.

To Dharma, Stacy and the little Lana.. my family in US.

To Alex.. the funniest and coolest prick i've ever met in my life.. love you, bro!

To Lucia, Daniele and Davide, Alisa and Cori, and to Giovanna.. it's not a long time i've been knowing you, but long enough to say you will always be good friends for me.

To all my relatives.. for having supported me in every difficult moment; for having helped me in taking good decisions where that was not so easy.

To Luisa (last in the list, but first in my heart).. you're the light breaking in my darkness, my shelter from the storm.. God ain't got me enough words for you.

To myself.. for doing it.. and more, much more than this: for doing it my way!! never forget you're a nobody without all these guys! never forget you still have a lot to learn! never forget who you are!

To You.. if you're reading this, and you're not named before, for sure you deserve my thanks, and i'm a dick-head for having forgotten you!! love you much.