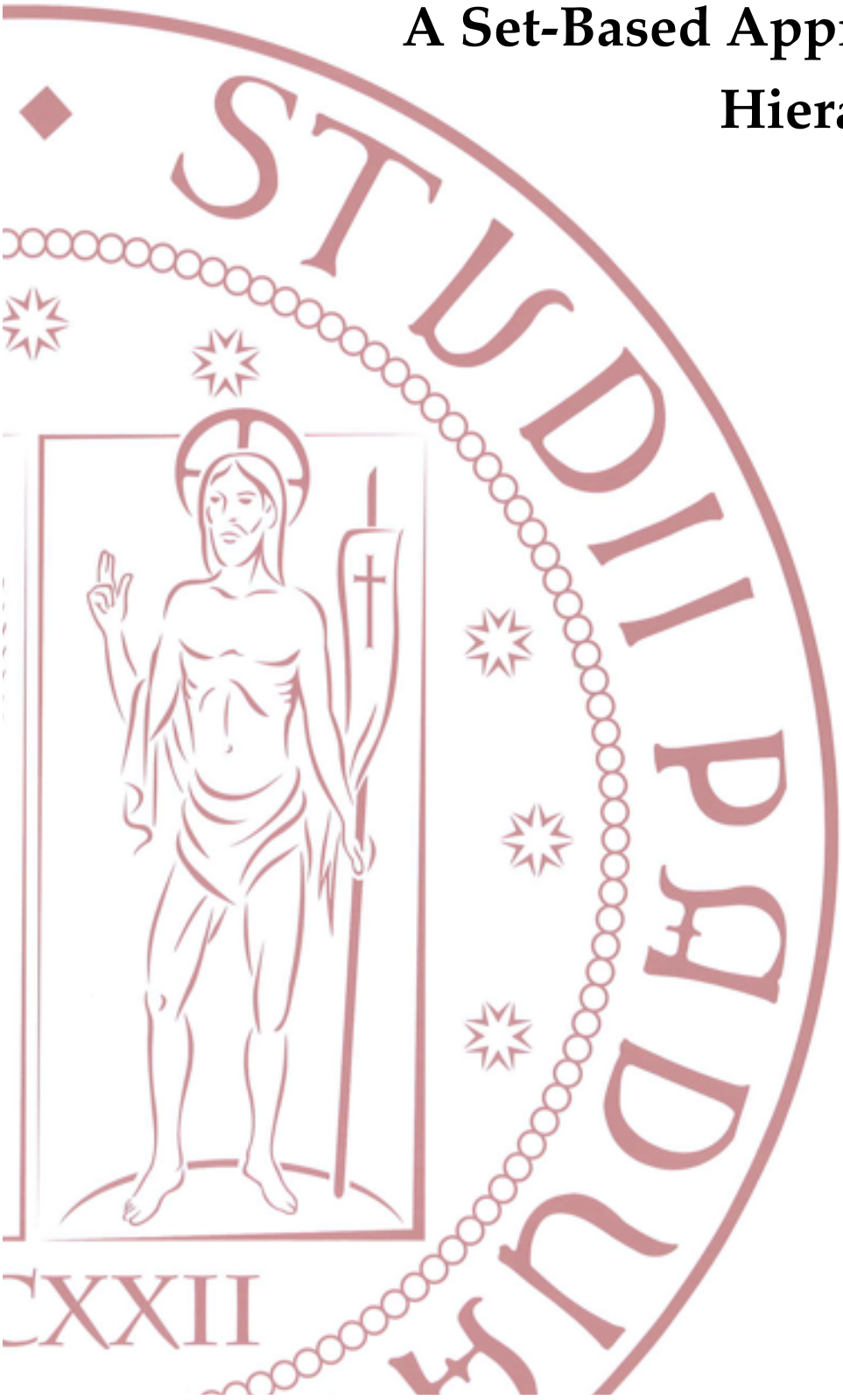
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DEI
DIPARTIMENTO DI
INGEGNERIA DELL'INFORMAZIONE

SCUOLA DI DOTTORATO IN INGEGNERIA DELL'INFORMAZIONE
INDIRIZZO IN SCIENZA E TECNOLOGIA DELL'INFORMAZIONE
CICLO: XXIII

A Set-Based Approach to Deal with Hierarchical Structures



Direttore della Scuola
PROF. MATTEO BERTOCCO

Supervisori
PROF. MARISTELLA AGOSTI
DR. ING. NICOLA FERRO

Dottorando
ING. GIANMARIA SILVELLO

28 GENNAIO 2011

Abstract

Hierarchical structures are pervasive in computer science because they are a fundamental means for modeling many aspects of reality and for representing and managing a wide corpus of data and digital resources. One of the most important hierarchical structures is the tree, which has been widely studied, analyzed and adopted in several contexts and scientific fields over time. Our work takes into major consideration the role and impact of the tree in computer science and investigates its applications starting from the following pivotal question: *"Is the tree always the most advantageous choice for modeling, representing and managing hierarchies?"* Our aim is to analyze the nature and use of hierarchical structures and determine the most suitable way of employing them in different contexts of interests.

We concentrate our work mainly on the scientific field of Digital Libraries. Digital Libraries are the compound and complex systems which manage digital resources from our cultural heritage belonging to different cultural organizations such as libraries, archives and museums and which provide advanced services over these digital resources. In particular, we point out a focal use case within this scientific field based on the modeling, representation, management and exchange of archival resources in a distributed environment. We take into consideration the hierarchical inner structure of archives by considering the solutions proposed in the literature for modeling, representing, managing and sharing the archival resources. Archives are usually modeled by means of a tree structure; furthermore, the standard de facto for digital encoding of digital cultural resources described and represented by means of metadata is the *eXtensible Markup Language (XML)* that supports a tree representation. The problem often affecting this approach is that the model used to represent the hierarchies is bounded by the specific technology of choice adopted for its instantiation e.g. the XML. In the archival context the tree structure is commonly instantiated by means of a unique XML file which mixes up the hierarchical structure elements with the content elements, without a clear distinction between the two; it is then not straightforward to determine how to access and exchange a specific subset of data without navigating the whole hierarchy or without losing meaningful hierarchical relationships.

To address the problems exemplified in the previous scenario we propose the *NEsted SeT for Object hieRarchies (NESTOR) Framework* which is composed of two main components: the *NESTOR Model* and the *NESTOR Prototype*.

The *NESTOR Model* is the core of the NESTOR Framework because it defines the set data models on which every component of the framework relies. It defines two set data models that

we have called the *"Nested Set Model (NS-M)"* and the *"Inverse Nested Set Model (INS-M)"*. We formally define these two set data models by showing how we can model and represent hierarchies throughout collections of nested sets. We show how these models add some features with respect to the tree while maintaining its full expressive power. We formally prove several properties of these models and show the correspondences with the tree. Furthermore, we define four distance measures for the the NS-M and the INS-M and we prove them to be metric spaces.

The NESTOR Model is presented from a formal point-of-view and then envisioned in a practical application context defined by the *NESTOR Prototype*. In order to describe the prototype we rely on the archive use case, and propose an application for modeling, representing, managing and sharing of archival resources. The expressive power of the archive modeled by means of a tree and the set data models are compared. We analyze the advantages and disadvantages of our approach when data management and exchange in distributed environments have to be faced. We provide a concrete implementation of the described models in the context of the informative system called *SIAR (Sistema Informativo Archivistico Regionale)* that we designed and developed for the management of the archival resources of the Italian Veneto Region. Furthermore, we show how the NESTOR Framework can be used in conjunction with well-established and widely-used Digital Libraries technological advances.

Sommario

Le strutture gerarchiche sono largamente utilizzate in informatica in quanto mezzi fondamentali per modellare molti aspetti della realtà e per rappresentare nonché gestire una grande varietà di dati e risorse digitali. Una delle strutture gerarchiche più rilevanti è l'albero, una struttura ampiamente studiata, analizzata e adottata in diversi contesti scientifici.

Questo lavoro tiene in grande considerazione il ruolo e l'impatto della struttura ad albero in informatica e ne studia le applicazioni a partire dalla seguente domanda: *"La struttura ad albero è sempre il modo più vantaggioso per modellare, rappresentare e gestire le gerarchie?"* A tal proposito ci si propone di analizzare la natura e l'uso delle strutture gerarchiche e di determinare il modo più adeguato per impiegarle nei diversi contesti considerati. Il lavoro si focalizza principalmente sul campo scientifico delle biblioteche digitali, ossia su quei sistemi composti e complessi che gestiscono risorse digitali culturali, provenienti da organizzazioni differenti quali biblioteche, archivi e musei e che definiscono servizi avanzati su tali risorse.

In particolare, il caso d'uso principale su cui è basato questo lavoro è la modellazione, la rappresentazione, la gestione e la condivisione delle risorse di tipo archivistico in ambiente distribuito. Si considera quindi la struttura gerarchica degli archivi tenendo presente le soluzioni per modellare, rappresentare, gestire e condividere le risorse archivistiche allo stato dell'arte. Gli archivi sono solitamente modellati mediante una struttura ad albero; inoltre, lo standard di fatto per la codifica delle risorse culturali digitali descritte e rappresentate per mezzo di metadati è l'*eXtensible Markup Language (XML)* che supporta la rappresentazione ad albero.

Nel contesto archivistico la struttura ad albero è comunemente istanziata mediante un unico file XML che, per come è definito e comunemente utilizzato, mischia elementi strutturali e di contenuto senza permettere una chiara distinzione tra di loro. Di fatto, non è facilmente definibile l'accesso e lo scambio di sottoinsiemi di dati e risorse senza costringere alla navigazione di tutta la gerarchia o senza perdere le significative relazioni gerarchiche tra le varie componenti.

Per affrontare i problemi esemplificati nel precedente scenario si propone il *NEsted SeT for Object hieRarchies (NESTOR) Framework* composto da due componenti principali: Il *NESTOR Model* e il *NESTOR Prototype*.

Il *NESTOR Model* è il nucleo del framework perché definisce i modelli ad insiemi su cui ogni componente del framework si basa. Il NESTOR Model definisce due modelli dati basati su insiemi annidati: il *"Nested Set Model (NS-M)"* e l'*"Inverse Nested Set Model (INS-M)"*. Questi modelli sono definiti formalmente mostrando come sia possibile modellare e rappresentare gerarchie mediante collezioni di insiemi annidati. Si dimostra come questi modelli ag-

giungano alcune caratteristiche rispetto alla struttura ad albero, mantenendone il potere espressivo. Si dimostrano formalmente diverse proprietà dei modelli ad insiemi e si mostrano le corrispondenze con la struttura ad albero. Si definiscono inoltre quattro misure di distanza per il NS-M e l'INS-M dimostrando come questi possano essere definiti come spazi metrici.

Il NESTOR Model è presentato da un punto di vista formale e successivamente adottato in un concreto contesto applicativo definito dal *NESTOR Prototype*. Per descrivere il NESTOR Prototype ci si basa sul caso d'uso degli archivi, proponendo un'applicazione per modellare, rappresentare, gestire e scambiare le risorse archivistiche. Il potere espressivo degli archivi modellati mediante l'albero e modellati mediante i modelli dati ad insiemi sono confrontati. Si analizzano i vantaggi e gli svantaggi dell'approccio presentato quando deve essere affrontata la gestione e lo scambio delle risorse in ambiente distribuito. Si fornisce un'implementazione concreta dei modelli descritti nel contesto del sistema informativo chiamato *SIAR (Sistema Informativo Archivistico Regionale)*, progettato e sviluppato per la gestione delle risorse archivistiche della Regione del Veneto. Inoltre, si mostra come il NESTOR Framework possa essere usato congiuntamente alle tecnologie definite e ampiamente utilizzate nel contesto delle biblioteche digitali per superare i problemi evidenziati dalle soluzioni allo stato dell'arte.

Contents

List of Figures

Chapter 1

Outline

The thesis is organized as follows:

Chapter 2: Introduction to Information and Data Structures. In this chapter we analyze the nature and use of hierarchical information and data structures in order to settle a common basis to envision suitable ways of employing them in different applicative environments. We describe different kinds of hierarchy pointing out the multifaceted nature of this concept. Furthermore, we analyze the use of the concept of hierarchy also from a historical point-of-view in order to highlight its complex and compound nature. Afterwards, we analyze the tree structure within several applicative environments in which it is employed.

Chapter 3: Overview of the Work. In this chapter we present the motivations of this work. We introduce the NESTOR Framework composed by the NESTOR Model and the NESTOR Prototype detailing the main innovations brought about by these components.

Chapter 4: Mathematical Background. In this chapter we present the mathematical concepts on which the NESTOR Framework relies. First-of-all, we describe the basics of set-theory focusing on the concepts of family of sets and ordered sets. Then we detail the aspects of graph theory which are useful for this work; in particular, we analyze the tree, its extensions and its metric properties.

Chapter 5: Digital Libraries. In this chapter we present the context in which this work is carried on. We present the main characteristics of Digital Libraries, in particular highlighting the aspects related to interoperability. We describe our main use case which is based of the archives; we detail the nature of archives and archival resources pointing out their characteristics and peculiarities. Lastly, we analyze several standard digital library technologies focusing on the Dublin Core Metadata Initiative, the Encoded Archival Description (EAD), the Open Archive Initiative - Protocol for Metadata Harvesting (OAI-PMH) and the Open Archive Initiative - Object Re-use and Exchange (OAI-ORE).

Chapter 6: The NESTOR Model. In this chapter we present the NESTOR Model giving the formal definitions of the two set data models called *Nested Set Model* (NS-M) and *Inverse Nested Set Model* (INS-M) on which it is based. Afterwards, we present the set-theoretical properties of the set data models and the formal relationships between them and the tree. We define two extensions of the set data models and lastly, we define and prove their metric properties.

Chapter 7: The NESTOR Prototype and its Implementation in the SIAR System. In this chapter we present the NESTOR Prototype which provides an instantiation of the NESTOR Model. We describe how archives and archival resources can be modeled and represented by means of the NESTOR Model and we point out two possible applications. The first application is the state-of-the-art for representing, managing, accessing and exchanging archival resources and the second is a solution which relies on the newly defined set-based approach. We compare these two applications pointing out advantages and disadvantages of both the presented solutions. Lastly, we describe the *Sistema Informativo Archivistico Regionale (SIAR)* which is a digital archive system designed and developed on the basis on the NESTOR Model and which employs the proposed set-based application.

Chapter 8: Conclusions. In this chapter we draw some conclusions and discuss future research directions.

Chapter 2

Introduction to Information and Data Structures

The aim of this chapter is to analyze the nature and use of hierarchical information and data structures in order to settle on a common basis to envision suitable ways of employing them in different contexts of interests. We analyze the use of hierarchies and hierarchical organizations in several scientific and non-scientific fields; in particular, we focus on the tree structure which represents the most important hierarchical structure in mathematics and computer science.

2.1 Hierarchy

What is meant by hierarchy? The term hierarchy is polysemous indeed, it can be defined [Hanks, 1979] as *"a system of persons or things arranged in a graded order"*, as *"a body of persons in holy orders organized into graded ranks"*, as *"the collective body of those so organized"*, as *"a series of ordered groupings within a system, such as the arrangement of plants and animals into classes, orders, families, etc."*, and as *"government by an organized priesthood"*.

Hierarchical structures are part of our common experience, in the physical as well as in the living and social worlds. Analyzing hierarchical structures in several environments we can see that it is difficult to attempt a unique and comprehensive definition. There are distinct levels in nature between the infinitely small – e.g. atoms or elementary particles – and the infinitely large – e.g galaxies and universe; there is a hierarchy of living entities, from cells to organs, organisms, species, populations and ecosystems; many social organizations like firms or administrative services display pyramidal layouts defining unequal degrees in power or competence between the levels; differences in size, wealth and power establish a hierarchical order between cities in a region or between countries in the world; various distinctive attributes define hierarchies of social status among groups or individuals in many societies. What we observe or think is never limited to a single scale of facts, it is not made up of an amorphous distribution of elements belonging to a single scale, but it is instead "*organized into more or less distinct levels that define a variety of scales in space and time and can in some cases be*

(a) Matryoshka

(b) Chinese Boxes

Figure 2.1: Two inclusion hierarchies.

ordered according to these scales" [Pumain, 2006].

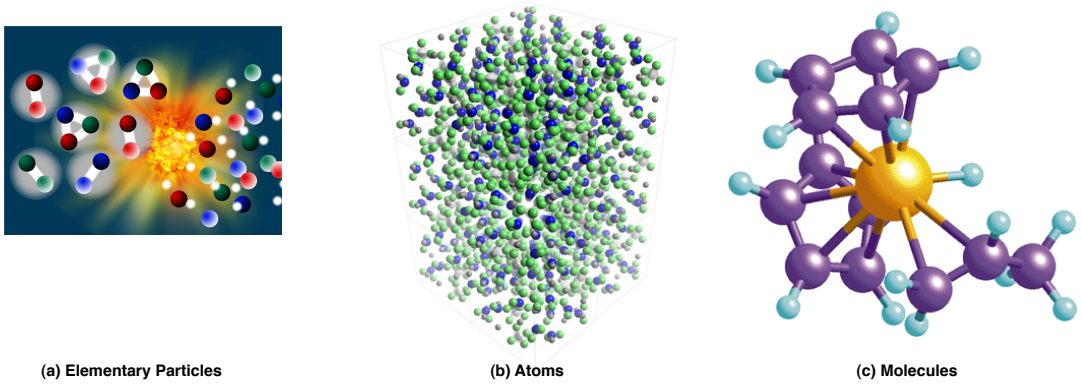
Hierarchical organizations are very frequent in natural and social systems that we observe; this can be explained by considering different hypotheses, which constitute a common ground for analyzing this complex concept. We may point out that hierarchies are our way of perceiving and understanding our environment; thus, the concept of hierarchy can be considered as an abstraction to model reality. We can also associate different meanings to the concept of hierarchy: when applied to a social organization, it relates to a pyramidal configuration, entailing a top-down or a bottom-up circulation of control and information. When applied to an open system, the elements are grouped into subsystems, according to a nested or inclusion relationship, or through a regular differentiation among the subsystems classified by size. In this section we explore the concept of hierarchy under different points-of-view in order to unravel its importance in several scientific and non-scientific fields.

2.1.1 Hierarchical Organizations

In order to conduct a systematic analysis of hierarchical organizations we analyze the different types of hierarchy that we may encounter in different application environments. We take into account the considerations and results of the analysis conducted in [Lane, 2006], where four main different kinds of "hierarchy" have been defined.

Order hierarchy: this is associated to be equivalent to an ordering induced by the values of a variable defined on some set of elements. An order hierarchy does not refer to relationships and interactions between the entities that comprise the hierarchy, much less give any role to hierarchy in conditioning entity relationships and interaction structures. Order is essential to hierarchy, but order alone is not what makes hierarchy important for complex systems, nor is hierarchy the source of the order in complex systems.

Inclusion hierarchy: this is associated with a recursive organization of entities. A popular way to see an inclusion hierarchy is through the "chinese box" – or matryoshka – metaphor (Figure 2.1): "[...] *hierarchy means a set of Chinese boxes of a particular kind. A set of*



(a) Elementary Particles (b) Atoms (c) Molecules

Figure 2.2: Elementary Particles – Atoms – Molecules

Chinese boxes usually consists of a box enclosing a second box, which in turn, encloses a third – the recursion continuing as long as the patience of the craftsman holds out. The Chinese boxes called hierarchies are variants of that pattern. Opening any given box in a hierarchy discloses not just one new box within, but a whole small set of boxes; and opening any one of these component boxes discloses a new set in turn" [Simon, 1962]. This kind of hierarchy can be seen as a special case of order hierarchy, where the ordering variable is the number of boxes one opens before arriving at the particular box of interest. However, it is more than this; it makes an ontological claim [Lane, 2006]. The boxes are seen as entities which contain other entities; this is a way to represent reality – at least the part of reality contained in an inclusion hierarchy – that follows a top down procedure. In this kind of hierarchy the entities at the lower levels are somehow constituent parts of the higher ones.

Control hierarchy: This is probably the most commonly used sense of hierarchy, particularly in reference to social organizations, and has to do with "who gives orders to whom". In this context, hierarchy refers to a control system in which every entity has an assigned rank and usually all power is concentrated in the entity at the top. In this sense a church, a political party or army can be organized as a control hierarchy.

Level hierarchy: This describes a particular kind of ontological organization, in which entities are posited to exist at different "levels". Each level is characterized by a particular spatio-temporal scale for its associated entities and for the processes through which the entities at this level interact with one another: the higher the level, the more extended the associated spatio-temporal scale [Lane, 2006]. Entities at a given level may, through their interactions, construct and maintain entities at higher levels, and higher-level entities may be, at least in part, composed of lower-level entities: these are often described by the term *upward causation*. Through upward causation a level hierarchy may compose an inclusion hierarchy; this is the case where higher entities are exclusively composed by lower entities. Level hierarchies are also formed by *downward causation*: incorporation into a higher-level entity can change the

properties and interaction modalities of lower-level entities[1]. However, even if in a level hierarchy the entities may be included into higher-level entities, they need not be: they are always regarded as *autonomous entities*.

We can point out several examples of order hierarchy in the field of mathematics; a meaningful example is represented by a sequence: an ordered list of objects (or events). An example of inclusion hierarchy is biological where we find: cells, tissues, organs and living organisms. Examples of control hierarchies can be the church or the army. As we have said the level hierarchies can be inclusion hierarchies when the higher entities are exclusively composed of the lower ones, such as: elementary particles – atoms – molecules (Figure 2.2) or cells – organs – individuals – species. A case in which a level hierarchy cannot be seen strictly speaking as an inclusion hierarchy is the case of organizational hierarchies such as: individuals – working groups or departments – firms – national economies.

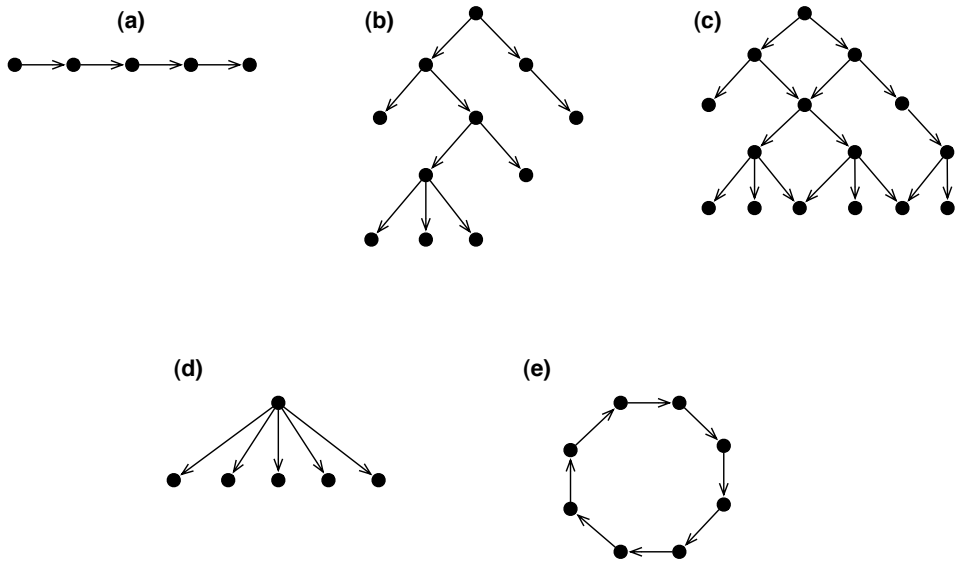


Figure 2.3: (a) Linear hierarchy. (b) Non-overlapping branching hierarchy (c) Overlapping hierarchy. (d) Not a hierarchy. (e) Shallow hierarchy. [Bateson and Hinde, 1976]

In Figure 2.3 we can see several graphs through which usually hierarchies are represented. In this representation the elements are depicted by black dots, and the relation between them by arrows. In Figure 2.3a we can see an example of what is called a linear hierarchy, that can be used to represent an order hierarchy. The model of a linear hierarchy can be associated to the concept of *chain* or *sequence*.

In Figure 2.3b we can see the representation of a hierarchy that has a tree structure where we have a top element which is superior to all the other elements in the hierarchy and at the

[1] An unusual point-of-view about upward and downward causations is given by the discussion between science and faith. Indeed, "*materialist scientists posit that there is no source of causality other than material interaction in this world*" [Goswami, 2008], and this is known as upward causation. Instead, "*all spiritual traditions agree that there is another causal power in the world. That causal power is what is called God*" [Goswami, 2008], this is known as downward causation.

same time every element can have at most one one direct superior element. In the following sections we will analyze in detail the tree which is our main focus.

In Figure 2.3c we find a kind of hierarchy which can be seen as an extension of the tree hierarchy. Indeed, we still have a top element but we can find elements which have more than one direct superior element. In mathematics this hierarchy is called a join-semilattice [Davey and Priestley, 2002].

In Figure 2.3d we can see a tree that is often called a shallow hierarchy because there is an element ruling over many sub-elements, but all the sub-elements are of equal status to each other. Lastly, in Figure 2.3e we can see a bunch of elements in a circular dependency and for this reason this is not defined as a hierarchy.

2.1.2 A Historical Viewpoint

The analysis of the concept of hierarchy in different application environments showed how this concept is extensive and hard to define. It is interesting to analyze the meaning of the concept of hierarchy from a historical point-of-view. The concept of hierarchy is hard to enclose within well-defined boundaries; "hierarchy" is polysemous and is a deep and powerful concept that, even from a historical point-of-view, has to be studied within a specific context of interest.

The use of word hierarchy dates back a considerable time. It seems to have been coined by *Pseudo-Dionysius the Areopagite* (6^{th} century AD) [Heil, 1991]. It is composed by *hieròs* which means "sacred" and *archía* from *àrchein* which means "rule". The first clear meaning arises from this etymology, since *hierarchy* at that time was "the governance of things sacred" [Verdier, 2006].

This meaning is strictly supported by the classical conception of the metaphysical order of the universe: "The Great Chain of Being" (Figure 2.4) in which all beings from the most basic up to the very highest and most perfect being are hierarchically linked to form one interconnected whole. The philosophical formulation of this notion is attributed to Plato even if its beginning is often associated with Aristotle[2], although it was never formulated until Plotinus's idea of universe. The universe is to be explained as a hierarchy of realities, a great chain of being, in which the higher reality is the cause of and gives existence to whatever is immediately below it; this process Plotinus calls emanation, an "overflowing" from a higher to a lower level. In this process there is a gradual diminution, so that every existent is slightly inferior to its cause.

This meaning of the word "hierarchy" persisted over time and it is emphasized as a theological term; in this context it is used to the "*subordination that exists between the different chorus of angels. There are nine choruses of blessed spirits divided into three hierarchies*" [Furetière, 1690]. The concept of hierarchy then enters the register of the description of the ecclesiastic state, and more generally that of society overall. The "deistic" view of the concept of hier-

[2]Aristotles "Great Chain of Being", "the Scala Naturae", or also referred to as the "Ladder of Life" was an early precursor of evolution theory, claiming that species can be ordered from the lowest to the highest, putting minerals and plants at the low end and humans on the highest end of the scale.

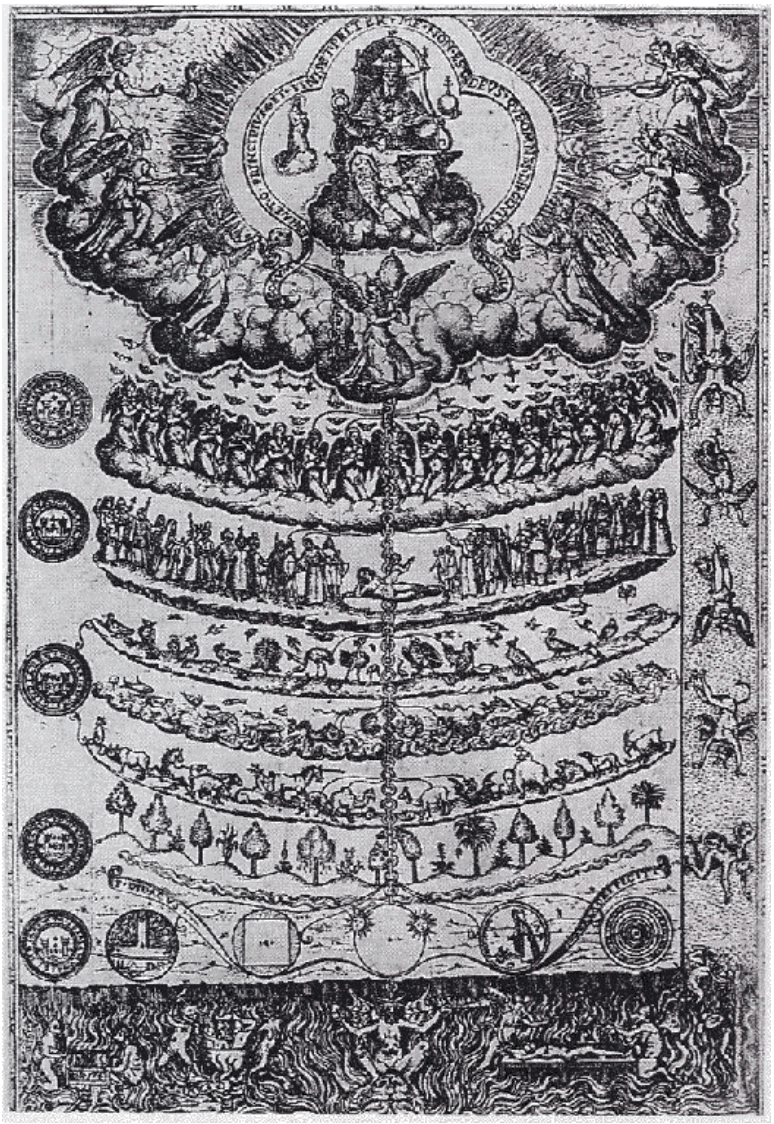


Figure 2.4: 1579 drawing of the Great Chain of Being from Didacus Valades, *Rhetorica Christiana*.

archy remained the predominant meaning until the "*Enclyclopèdie*" produced by Diderot and d'Alambert where hierarchy appears as an essentially human construction: "[...] *in civil society there are different orders (ranks) of citizens rising one above the other, and the general and particular administration of things is distributed in portions to different men or classes of men, from the sovereign who rules everyone down to the mere subject who obeys*" [Verdier, 2006].

This definition of hierarchy opened up a new way of seeing the concept of hierarchy widening its application; indeed, the notion of hierarchy widened from the side of principle to the side of applications. A very meaningful example is the use of the concept of hierarchy in natural sciences, as for instance in Linnaeus and the taxonomy of organisms. In the taxonomy of Linnaeus there are three kingdoms divided into classes, which in turn are divided into orders, genera, and species, with an additional rank lower than species. The taxonomy of Linnaeus is an example where the very concept of hierarchy is practically adopted even though the word of

"hierarchy" is not formally used. Another example came from Georges-Louis Leclerc, Comte de Buffon who wrote about the imperceptible evolution of one being into another and theorized that life had originated in a single center, and had spread outwards. These are not the only instances that can be called upon. Indeed, it is clear that these descriptions of nature served as models – or even metaphors – for numerous descriptions of society. In 1835 the "*Dictionnaire*" of the "Académie Française" extended the concept of hierarchy by relating the definition pertaining to the angels and the clergy to other sorts of power, authority, and rank subordinated one to another: political hierarchy, hierarchy of power, military hierarchy, etc.

From the above set of examples we can see that the concept of hierarchy is polysemous; there are several variations of the concept and the ways it has been used throughout history. From a historical point-of-view, it is possible to attempt to define of four main variations of the use of concept of hierarchy [Verdier, 2006] which are the:

1. Range of the concept.
2. Continuity and discontinuity of the concept.
3. Status given to the concept.
4. Nature of the concept.

The first point is that of the *range of the concept*: in some instances "hierarchy" can explain the whole universe, as with Milton who refers to "*the scale of nature set from centre to circumference*" [Milton, 1674]; indeed, in *Paradise Lost* [Milton, 1674] the inter-relationship of the spheres is seen as a relationship of "alimental" dependency in which the coarser elements enable the purer elements to exist. Or it can explain society as a whole, as with Durkheim: "*It is because human groups fit one into the other* [...] *that groups of objects are set out in the same order. Their regularly decreasing extension as we pass from genus to species, from species to variety, arises from the likewise decreasing extension presented in social divisions as one moves away from the widest and most ancient towards the most recent and derived. If all things are conceived as a single system, it is because society is conceived in the same manner. It is* [...] *the single whole to which all else is referred. Thus logical hierarchy is merely another aspect of social hierarchy and unity of knowledge is nothing other than the very unity of the community, extended to the universe*" [Durkheim and Mass, 1902]. Conversely the range of the concept can be limited to a specific object as with Marx who uses it solely to describe the military system: the workers, "*simple soldiers of industry are placed under the supervision of a complete hierarchy of officers and commanders*" [Marx and Engels, 1948].

The second point is the contrast between *continuity and discontinuity*. From this point-of-view the contribution of the article "*Group Psychology and the Analysis of the Ego*" by S. Freud is fairly relevant; he compared the two hierarchies of the Church and the army. In the former case we have Christ who loves all men, and men who must love one another in order to love Christ: the two levels are essentially different [Verdier, 2006]. In the latter case, there is a descending relationship from one level to the next which ensures "transfer of the libidinal

bond of fatherhood, without this bond changing in nature". The main difference between these two hierarchies is that in the Church all men are equal before Christ, since we have equal rights to his love; instead in the army hierarchy is made up of a hierarchy of successive formations. Thus, Freud presented two examples of essentially different hierarchies and established the foundations of the differentiation among hierarchies; in addition in this case we can find a change of the scale of observation by shifting from a family principle (fatherhood) to a social construction by hierarchy. Another important point is that in this context hierarchy is not an assumption but a product.

The third point to be made regards the *status given to the concept*. The concept of hierarchy can be used with different connotations; a meaningful example is "*La Démocratie en Amérique*" by Tocqueville published in France in 1835: the word hierarchy is used here in three ways, two negatives and one quite positive [Verdier, 2006]. The first concerns the social hierarchy of the aristocracy, the second the administrative hierarchy which "*produces perverse centralizing effects*" and the third concerns the natural social hierarchy which arises from the activities of the individuals and which in Tocqueville's opinion is necessary for civilization.

The final point concerns *the nature of hierarchy* which means the place given to hierarchy in reasoned thought. There are two opponents in this case: one which proposes the idea of a hierarchy which already exists and which has to be discovered as in the case of social Darwinism and the other in which hierarchies are elaborated by researchers of a given area as in the case of the definition given in the *Enclyclopèdie*.

2.2 Trees

The concept of hierarchy is complex and multifaceted and as common sense suggest it is straightforwardly associated to the concept of tree. A tree is a particular kind of hierarchy which respects some specific conditions. In Figure 2.3 we have seen several trees: Figure 2.3a, 2.3b and 2.3e.

Following the informal definition of hierarchy that we introduced in Section 2.1.1, we can say that a tree is a linear hierarchy or a non-overlapping branching hierarchy. Thus, a hierarchy is a tree if there are no loops (like in Figure 2.3f) or circuits (like in Figure 2.3d) and every element is not bossed by more than one element (like in Figure 2.3c and 2.3g). The tree is the most diffuse way to model a hierarchy and it is widely adopted in many disciplines ranging from biology to computer science and humanities.

In Figure 2.5 we can see one of the most famous trees in history and probably one of the first examples of representation of a hierarchy by means of this model. Figure 2.5 reproduces the only tree in the *"Origin of Species"* of Charles Darwin; it appears in the fourth chapter, "*Natural Selection*" in the section "*Divergence of character*". Darwin did not call it a tree when the image is first introduced: "*Now let us see how this principle of great benefit derived from divergence of character, combined with the principles of natural selection and of extinction, will tend to act. The accompanying diagram will aid us in understanding this rather perplexing*

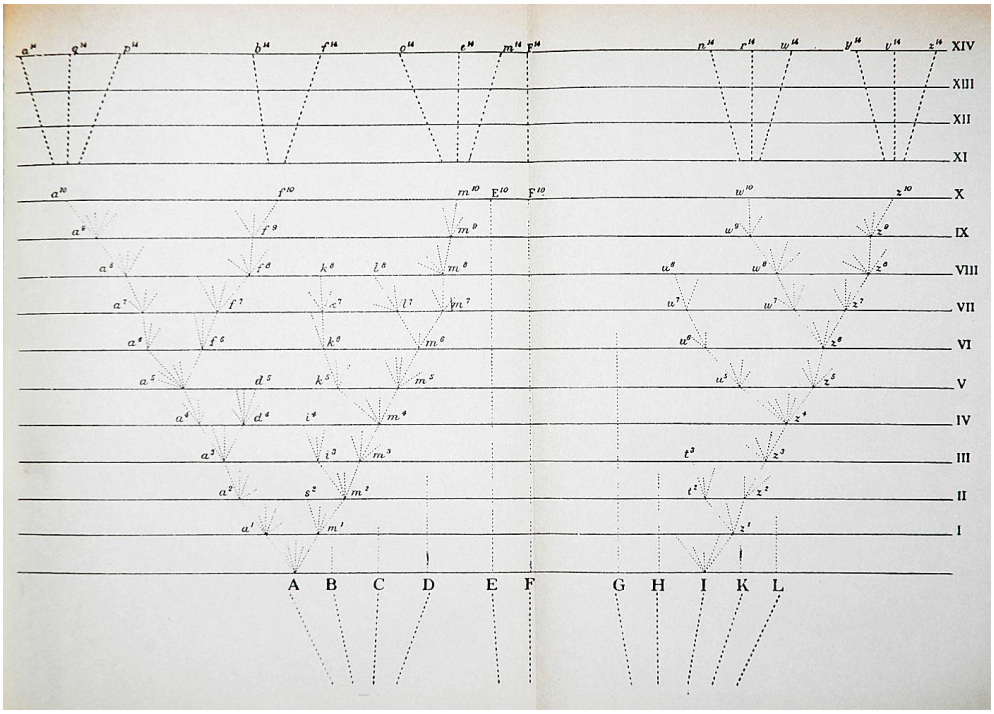


Figure 2.5: Divergence of character (*Origin of Species*, Charles Darwin).

subject . . .". The word "tree" appears only at the end of the chapter, and "*surrounded by signs of hesitation, possibly because of the religious echoes associated with the Tree of Life*" [Moretti, 2005]; however, Darwin wrote:"*The affinities of all the beings of the same class have sometimes been represented by a great tree. I believe this simile largely speaks the truth*" [Darwin, 1859].

The idea of the evolutionary tree had been in Darwin's mind for a long time; indeed, in Figure 2.6 we can see a famous tree representation taken from Darwin's notebook and dated 1837. Twenty-two years later, in the correspondence between Darwin and Murray (1859) we can find another hint about the novelty of the tree representation; indeed, Darwin wrote: "[. . . it is] *an odd looking affair, but is indispensable*" [Burkhardt and Smith, 2001].

An evolutionary tree constitutes a morphological diagram in which history is systematically correlated with form; for evolutionary thought morphology and history are truly the two dimensions of the same tree: "*where the vertical axes charts, from the bottom up, the regular passage of time, while the horizontal one follows the formal diversification that will eventually lead to well-marked varieties, or to entirely new species*" [Moretti, 2005].

From a single common origin, to an immense variety of solutions: it is this incessant growing-apart of life forms that the branches of a morphological tree capture with such intuitive force: "*A tree can be viewed as a simplified description of a matrix of distances*" [Cavalli-Sforza et al., 1994]. [Cavalli-Sforza et al., 1994] leads us to see another use of the tree in a different field of studies: linguistics. In Figure 2.7 we can see a "Linguistic tree" [Cavalli-Sforza, 1997] where its mirror-like alignment of genetic groups and linguistic families drifting away from each other makes clear that "*a tree is a way of sketching how far a certain language*

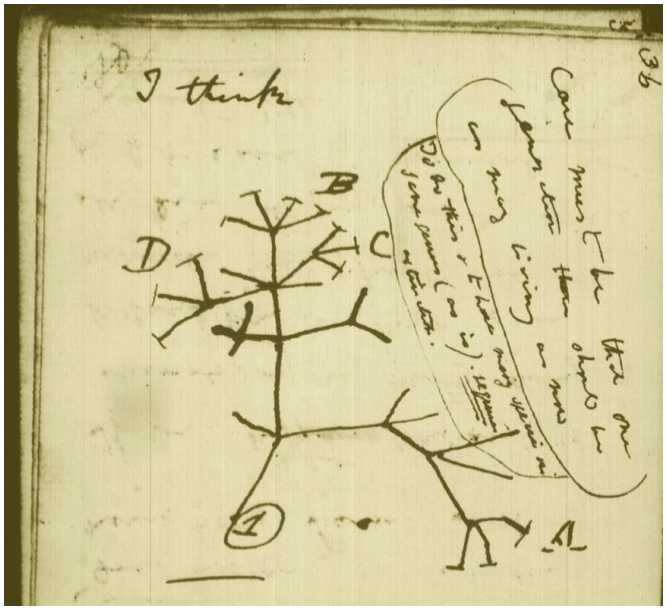


Figure 2.6: Tree of Life: the first-known sketch by Charles Darwin of an evolutionary tree describing the relationships among groups of organisms. (Syndics of Cambridge University Library).

has moved from another one, or from their common point of origin" [Moretti, 2005]. This vision suggests the idea of metrics associated with a tree; somehow establishing "how far" a language moves from another one indicates the presence of a distance measure, which is a very important concept as we will see.

The tree was born as a mere representation of a particular kind of hierarchy and has evolved becoming an indispensable structure to model reality. Furthermore, the tree structure has been formalized and several properties have been pointed out helping cross-field scientists to model and then find a solution to their scientific issues. A relevant example is the *phylogenetic tree* which originates directly from the evolutionary tree in Figure 2.5. That is a tree which represents the divergence of a trait over time, or the distance of the trait in question from the moment of its initial observation to that of its final observation after a certain lapse of time. To explain this observation in a clearer way and to introduce some basic terminology and concepts about trees we present a basic method for defining a phylogenetic tree which follows a meaningful example originally presented in [Piazza, 2005].

So let us suppose we want to build a phylogenetic tree of a gene, and this gene is the sequence of DNA made up of five elements: AATTC. The gene evolves over time – that is it changes. To represent this change in the form of a tree, we have to define both a point of departure – called the "*root*" of the tree – and a line that joins the root to a point representing the moment at which the first change happens, which we can imagine as altering the third element from T to G – i.e. from AATTC to AAGTC. From now on the two genes AATTC and AAGTC coexist, and their coexistence is represented by a bifurcation of the tree – this point is called a "*node*" of the tree. Every successive episode of diversification can be represented analogously by a node at which a bifurcation originates, and the tree becomes a sequence of bifurcations, at

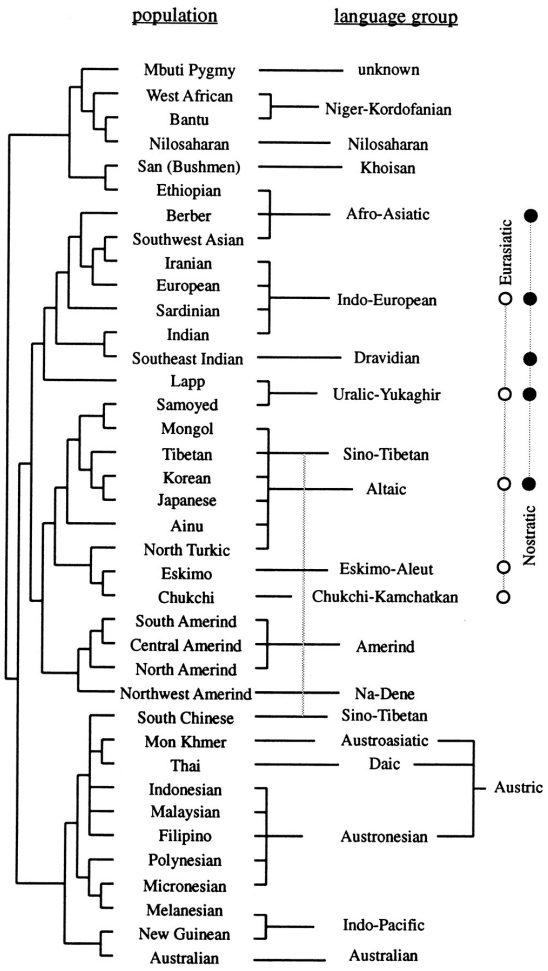


Figure 2.7: Linguistic tree (image originally published in [Cavalli-Sforza, 1997]).

the end of which we have as many genes that have been generated from the original in the root of the tree. The elements at the extremities of the tree are technically called *leaves*. In [Piazza, 2005] the author pointed out that in a real tree of life we do not have only bifurcations but trifurcations, quadrifurcations and so on and so forth.

The definition and the use of trees to model such aspects of reality allow us to use this structure to define problems and seek solutions. In the phylogenetic context, as an example, a well-known problem is to infer an evolutionary tree from a set of measurements; this particular problem crops up in various fields, such as biology, paleography, and archeology [Buneman, 1971]. For examples, amino-acid sequences of the same protein extracted from different organisms can be determined, and from the dissimilarities between these sequences one can attempt to construct a phylogenetic tree of these organisms [Buneman, 1971]. A similar situation occurs when one has a set of manuscripts all directly or indirectly copied from a common original manuscript: "*One seeks to reconstruct a family tree or stemma of these documents from errors*" [Buneman, 1971]. A frequent starting point in the solution of such a problem is the measurement of a *dissimilarity coefficient* between the vertices of the tree. We can see that in

the treatment of such a problem we move from a pure graphical representation of a hierarchy – e.g. the evolutionary tree of Figure 2.6 – to a mathematical treatment of the tree where we can define properties between the nodes and try to use them to define and solve problems.

2.2.1 Phylogenetics and Graph Theory: A Step Towards Formalization

Since Darwin, a deep ongoing problem in biology has been to determine the history of the evolution of species that has brought us to our current state. We have seen that the tree is considered an essential structure for this purpose and it has been adopted and exploited in many fields other than biology. The great utility of the tree lies in the possibility to mathematically formalize it, define properties over it and then exploit this structure to model reality, formally define problems and look for solutions. In this section we introduce some basic concepts that lead us to a mathematical view of the tree giving us a hint about how its mathematical properties can be helpful for solving problems. Throughout this still rather informal treatment we consider phylogenetics as an aid to help explain some tree related concepts.

It is natural when we think about these concepts to draw a graph so we can model the reality we are considering. A graph is a mathematical structure that consists of some elements called *vertices*, some pairs of which are deemed to be "joined" or "adjacent" [Gowers et al., 2008]. It is customary to represent the vertices by points in a plane and to join adjacent points by a line – e.g. Figure 2.3. The line is referred to as an *edge*. For example, the rail network of a country can be represented as a graph: the vertices are the stations, and every edge between two vertices represents a rail track from two stations. Another example is provided by the Internet: every computer is a vertex and an edge represents a direct link between two computers. Many questions in graph theory take the form of asking what some structural property of a graph can tell us about its other properties. A classical example is trying to find a graph with *n* vertices that does not contain a triangle – i.e. a set of three vertices mutually joined; in this case a classical problem is: *How many edges can the graph have?* This question arises in several application and optimization problems such as the Travel Salesman Problem.

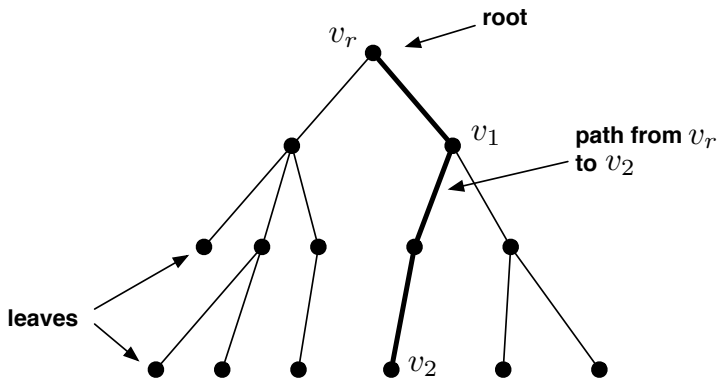


Figure 2.8: A rooted tree [Gowers et al., 2008].

When we think of a phylogenetic tree we can draw a graph where the set of vertices – call

it V – are species and an edge from species v_1 to species v_2 indicates that v_2 evolved directly from v_1. To explain some mathematical issues we consider a simple special case represented in Figure 2.8. In this graph we can distinguish the root that we indicate with v_r; this tree is then called *rooted tree* because we can point out this particular node. Notice that, because there are no cycles, there is exactly one path in the tree from v_r to each vertex v. We say that $v_1 \leqslant v_2$ if the path from v_r to v_2 contains v_1 (see Figure 2.8).

A classical problem in phylogenetics is to determine which trees with a given set of leaves X (current species) and a given root vertex v_r (a hypothesized ancestral species) are consistent with experimental information and theoretical assumptions about the mechanism of evolution. Such a tree is called *rooted phylogenetic X-tree* [Gowers et al., 2008]. One can always add extra intermediate species, so typically one imposes the additional restriction that the phylogenetic trees be as simple as possible.

Suppose that we are interested in a certain characteristic, number of teeth, for example. We can use it to define a function f from X, the set of current species, to the nonnegative integers: given a species x in X, we let $f(x)$ be the number of teeth of members of x. In general a *character* is a function from X to a set C of possible values of a particular characteristic – e.g. having or not a particular gene, the number of vertebrae, the presence or absence of a particular enzyme, etc. [Gowers et al., 2008]. It is characters such as these that are measured by biologists in current species.

Another example that points out the importance of defining mathematical properties within the tree is represented by the concept of *distance* related to evolutionary steps; in our formulation of tree, we have treated the edges alike when in fact some may represent longer or shorter evolutionary steps. Suppose that we have a function w that assigns a positive number to each edge. Since there is a unique shortest path between any two vertices in the tree, w induces a distance function $\mathrm{d_w}$ on the vertices. This distance measure can be used to determine the dissimilarity coefficient that we introduced in the previous section; this distance in literature is called *additive tree metric*. The mathematical treatment of the tree conducted us to the definition of this widely-used property that in phylogenetic trees is used to try to describe the relations between contemporary species; e.g. the distance between two species can be determined by the number of different sequences for some protein or for the DNA.

The definition of distance or additive tree metric has been useful also in several other fields such as cognitive psychology [Abdi, 1990]. In cognitive psychology the additive tree metric has been used to study the so-called *typicality effect*. This term is used by psychologists to express that some members of a natural category are more representative of that category than are some of the other members. The classical example [Abdi, 1990] is that for the category "bird", "canary" better represents the category than " penguin". As such, class elements are not equivalent and are often more or less representative of their category. Clearly, the standard representation of a tree where edge length – i.e. a metric imposed on the tree – is not considered forces objects to be equivalent. The possibility of defining an additive tree metric on a tree allows the psychologists to adopt this model and to exploit its mathematical properties to solve

a particular problem. Indeed, in [Tversky, 1977], the tree representation is considered more than a simple tool; it has been shown that an additive tree distance is a particular case of a more general model of similarity that allows for stimulus similarity to be measured in terms of both common and distinctive features. In [Sattath and Tversky, 1977] we can see that similarity data can be represented as a tree with a super-imposed additive metric; some empirical and theoretical advantages of tree representation are illustrated where one of the most interesting outcomes is the definition of a computer algorithm for the construction of additive similarity trees.

Lastly, we point out that the mathematical properties of trees were first exploited in the field of chemistry to study the chemical problems of isomers [Cayley, 1891] by successfully using "*recursion formulas for counting the number of trees having a finite number of vertices, where the number of branches at a vertex was not limited*" [Otter, 1947]. This problem was further studied by [Henze and Blair, 1931] who developed a recursion formula for counting the number of trees having the same finite number of vertices; this was the first solution to a problem of isomers in chemistry. Afterwards, in 1937, G. Pólya developed a powerful method for treating the symmetries of certain types of geometrical configurations under a given permutation group. He implicitly exploited the recursion formulas by Henze and Blair, and Cayley "*solving several problems connected with chemistry*"[3] [Otter, 1947].

From these examples we can see that the mathematical treatment of trees allows us to define automatic procedures – i.e. algorithms – to construct and manipulate trees.

2.2.2 The Tree Data Model

The tree in computer science has a great and fundamental impact; its mathematical definition and its properties are the starting points for the definition of the tree data model and of important and widely-used algorithms developed over it. In [Knuth, 1997] Knuth said that: "[trees] *are the most important nonlinear structures that arise in computer science*". In this section we describe the tree in the context of computer science and define further terms and concepts related to them, and we provide some examples of algorithms that operate on them.

In computer science when we talk of a tree – if not explicitly stated – we refer to a rooted tree adopting the terminology and the – informal – definition given in Section 2.2.1 when we described phylogenetic trees. We have seen that a tree is a graph with several conditions imposed over it; the vertices of a tree in computer science are called *nodes*. This differentiation helps us to distinguish when we are talking of a general graph and when we are dealing with a tree.

We consider a tree an abstract data type that stores elements hierarchically [Goodrich and Tamassia, 2001]. With the exception of the top node – i.e. the root – each node in a tree has a *parent* node and zero or more *children* elements. Thus, a tree T is considered as a set of nodes storing elements in a *parent-child* relationship. The most diffuse definition of tree in computer

[3]The original paper by Pólya is in German [Pólya, 1937]; we refer to the citations by Richard Otter in [Otter, 1947].

science is recursive: we defined a tree in terms of trees; there are also non-recursive definitions, but a recursive one seems to be more appropriate because of the very recursive nature of the tree. Indeed, in [Knuth, 1997] a tree is formally defined as a finite set T of one or more nodes such that:

1. there is one specially designed node called the *root* of the tree, `root(T)`; and
2. the remaining nodes (excluding the root) are partitioned into $m \geq 0$ disjoint sets $T_1, \ldots, T_m$, and each of these sets in turn is a tree. The trees $T_1, \ldots, T_m$ are called *subtrees* of the root.

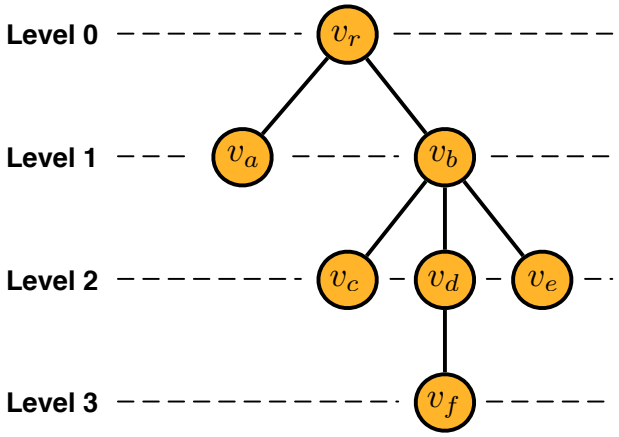


Figure 2.9: The levels of the nodes of a tree.

It follows from this definition that every node of a tree is the root of one subtree contained in the whole tree. The number of subtrees of a node is called *degree* of that node. The *level* of a node with respect to T is defined recursively: The level of `root(T)` is zero, and the level of any other node is higher than that node's level with respect to the subtree of `root(T)` containing it. The concepts are illustrated in Figure 2.9, which shows a tree T with seven nodes. The root of the tree is the node labeled v_r, and it has two subtrees $T_1 = \{v_a\}$ and $T_2 = \{v_b, v_c, v_d, v_e, v_f\}$; the tree T_2 has the node v_b as its root. Node v_b is at level 1 with respect to the whole tree, and it has three subtrees $T_3 = \{v_c\}$, $T_4 = \{v_d, v_f\}$, and $T_5 = \{v_e\}$: therefore v_c has degree 3. The leaves are the nodes v_a, v_c, v_f, v_e. We can say that the root v_r is the parent of both nodes v_a and v_b and vice versa that v_a and v_b are children of v_r; the node v_r is not the parent of the nodes v_c, v_d, v_e, and v_f but it is an ancestor of these ones, vice versa these nodes are descendants of v_r. We can see that the standard terms adopted for tree structures is taken from the terminology used in the context of *family trees*; for this reason we can also talk about *sibling nodes* – i.e. two or more nodes which have the same parent node.

Optionally, we can assign a left-to-right order to the children of any node. For instance, the order of the children of v_b in Figure 2.9 is v_c the leftmost, then v_d, then v_e. This left-to-right ordering can be extended to order all the nodes in a tree. v_a and v_b are siblings and as v_b is to

the right of v_a then all the descendants of v_b are to the right of v_a [Aho and Ullman, 1992]. So, when the order between relative subtrees of a node is taken into account, we talk of *ordered trees*. The very nature of computer representation defines an implicit ordering for any tree, so in most cases ordered trees are of greatest interest to computer scientists [Knuth, 1997].

In this context it is important to highlight another kind of tree which is the *labeled tree* [Aho and Ullman, 1992]. A labeled tree is a tree in which a label or value is associated with each node of the tree. We can think of the label as the information associated with a given node. The label can be something simple, such as an integer, or complex, such as the text of an entire document. We can change the label of a node, but we cannot change the name of a node. If the name of a node is not important, we can represent a node by its label. However, the label does not always provide a unique name for a node, since several nodes may have the same label.

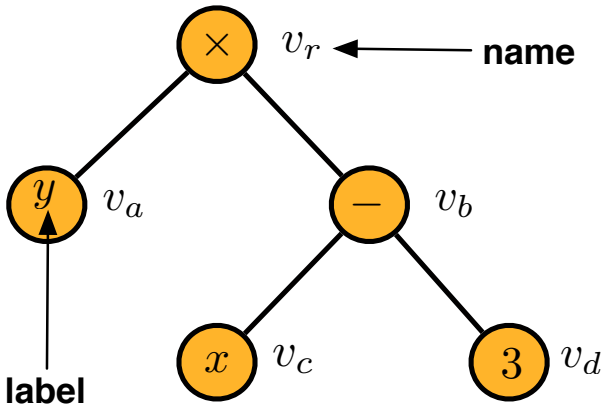


Figure 2.10: Expression tree for the arithmetic expression: $y \times (x - 3)$.

Thus, we can draw a node with both its label and its name as in Figure 2.10. The tree in Figure 2.10 is an *expression tree* which represents an arithmetic expression where the leaves are associated with variables and constants and the other nodes with the operators: inside the circles representing the nodes we put the label and at the right of the circle we put the name of the node.

2.2.3 The Tree Abstract Data Type

The tree abstract data type stores elements at positions, which, as with positions in a list, are defined relative to neighboring positions. The positions in a tree are its nodes, and neighboring positions satisfy the parent-child relationships that define a valid tree [Goodrich and Tamassia, 2001]. Therefore, it is possible to use the terms “node” and “position” interchangeably for trees.

The algorithms for performing computations on a tree are defined on the basis of several tree abstract data type basic methods. These methods can be divided into three main groups: *accessors*, *query* and *generic* methods. We present a significant subset of these methods with a clear understanding that this list is extensible.

The *accessor methods* return and accept positions, such as the following:

`root(T)` returns the root of the tree T.

`parent(v)` returns the parent of node v.

`children(v)` returns the list of the children of node v.

In the same way we can define methods that return the list of ancestors or descendants as well as the list of all the siblings of a node. The *query methods* allow us to verify the state of a node:

`isInternal(v)` tests whether node v is a branching node.

`isLeaf(v)` tests whether node v is a leaf.

`isRoot(v)` tests whether node v is the root.

There are a number of methods a tree should probably support that are not necessarily related to its tree structure [Goodrich and Tamassia, 2001]. Such generic methods for a tree include the following:

`size(T)` returns the number of nodes in the tree T.

`swapElements(v,w)` swaps the elements stored at position v and w.

`replaceElement(v,e)` replaces with e and returns the element stored at position v.

It is possible to extend this list of methods, for instance with some update methods or other access methods. We can see that these extensible lists of methods defined on a tree allow us to formally define algorithms and to study their running time. For instance, typical algorithms that we can define on trees are the *traversal algorithms*. A traversal of a tree T is a systematic way of accessing or visiting, all the nodes of T. The traversal schemes are divided into inorder (symmetric), preoreder and postorder traversal. In an inorder the tree is traversed starting form the left subtree, passing from the root and lastly visiting the right subtree. In a *preorder traversal* of a tree T, the root of T is visited first and then the subtrees rooted at its children are traversed recursively. If the tree is ordered, then the subtrees are traversed according to the order of the children. The specific action associated with the visit of a node v depends on the application of the traversal, and could involve anything from incrementing a counter to performing some complex computation for v. The preorder traversal algorithm is useful for producing a linear ordering of the nodes of a tree where parents must always come before their children in the ordering. Such ordering has several applications; for instance if we have a tree associated with a book chapter – see Figure 2.11 – preorder traversal allows us to examine the entire chapter sequentially, from beginning to end.

The *postorder traversal* can be seen as the inverse of the preorder, because it recursively traverses the subtrees rooted at the children of the root first, and then visits the root. The

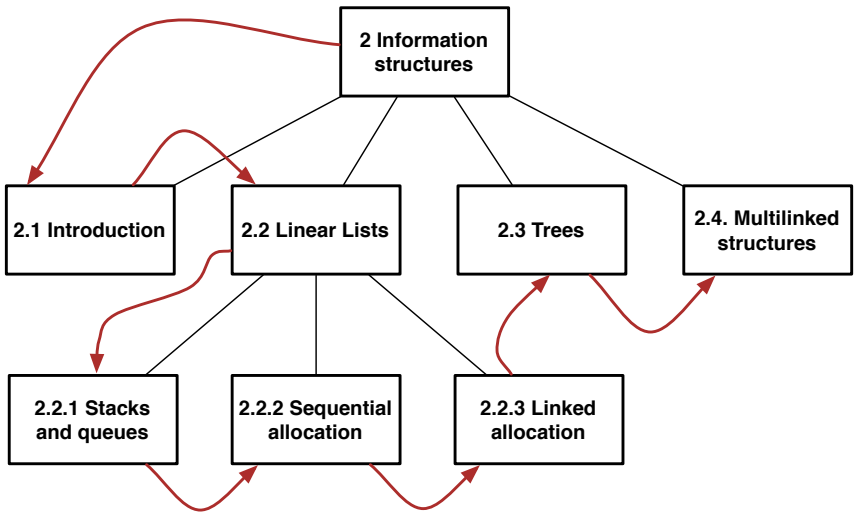


Figure 2.11: Preorder traversal of an ordered tree representing the structure of a book chapter [Knuth, 1997].

postorder traversal method is useful for solving problems where we wish to compute some property for each node v in a tree, but computing that property for v requires that we have already computed the same property for v's children. For instance, it can be used to visit the expression tree in Figure 2.10 to solve the expression the tree represents.

As a last example of a problem related to trees that can be addressed in an algorithmic way is the relevant problem of finding the *lowest common ancestor* between two nodes in a tree. The least common ancestor of two nodes v_i and v_j in a tree T is the node v_k that is an ancestor of both v_i and v_j and that has the higher level in T. The lowest common ancestor problem has been studied intensively both "*because it is inherently beautiful and because fast algorithms for the lowest common ancestor problem can be used to solve other algorithmic problems*" [Bender et al., 2005]. The problem of finding the lowest common ancestor has several concrete applications, for instance in phylogenetic trees where we may have to find the relationships between species and their lowest common ancestor. The lowest common ancestor problem was defined in [Aho et al., 1973], and a trivial way to solve it consists in finding the first intersection of the paths from v_i and v_j to the root of the tree[4]. However, much more efficient algorithms have been defined for instance in [Harel and Tarjan, 1984] where we can find the first algorithm that optimally finds the lowest common ancestor.

In Figure 2.12 we can see a graphical representation of the algorithm to find the lowest common ancestor v_k of the two nodes v_i and v_j. The path from v_i to v_r is colored in red and the one from v_j to v_r in blue; we can see that the intersection between these two paths is the path from v_k to v_r. In this path v_k has the higher associated level so it is the lowest common ancestor of v_i and v_j.

[4]The computational time required for this algorithm is $O(h)$ where h is the height of the tree [Bender et al., 2005] – i.e. the length of the longest path from a leaf to the root.

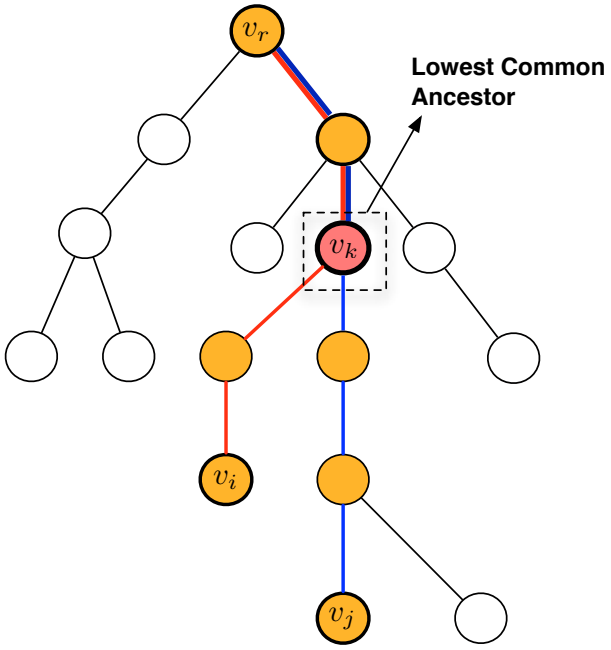


Figure 2.12: The graphical representation of the algorithm for finding the lowest common ancestor v_k of the two nodes v_i and v_j.

We have just presented several examples of how the mathematical definition of the tree model allows us to systematically treat the hierarchies modeled by means of it. In particular, a formal treatment of the tree data model allows us to:

- Model the reality of interest – e.g. evolutionary trees and phylogenetic trees.
- Define new data models that extend or restrict the tree data model and that better suit the reality of interest we need to model – e.g. rooted trees, ordered trees and labeled trees.
- Define and prove properties of the data model – e.g. additive tree metric.
- Design and develop algorithms on the data model – e.g. preoreder traversal, postorder traversal and the lowest common ancestor.

2.2.4 Graphical Representations

Trees can be drawn in many ways; when we think of a tree we usually draw it as a collection of dots and lines connecting them as we have done in the previous sections. In Figure 2.13 we can see three alternative representations of the same tree based on a collection of circles and lines; the main difference between these representations is the position of the root. As Knuth wrote in [Knuth, 1997]: "*It is not a frivolous joke to worry about how tree structures are drawn in diagrams, since there are many occasions in which we want to say that one node is above or higher than another node, or to refer to the rightmost element, etc. Certain algorithms for dealing with tree structures have become known as top down methods, as opposed to bottom*

up. Such terminology leads to confusion unless we adhere to a uniform convention for drawing trees."

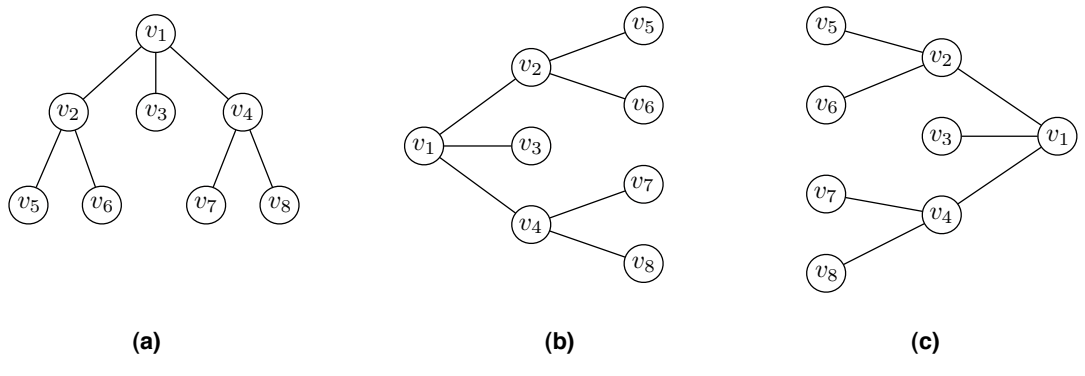


Figure 2.13: Alternative representation of the same tree by means of dots and lines.

Observing the computer literature we can see that the adopted convention is to draw the root at the top of the tree – e.g. Figure 2.13a. In this way we can talk about *shallow* and *deep* levels of a node and we can consistently adopt all the terminology derived from family trees: parent, children, ancestors, etc.

In [Knuth, 1997] alternative ways to graphically represent a tree have been proposed; these representations bear no resemblance with the ones we have considered till now. In Figure 2.14 we can see three alternative graphical representations of the tree in Figure 2.13a.

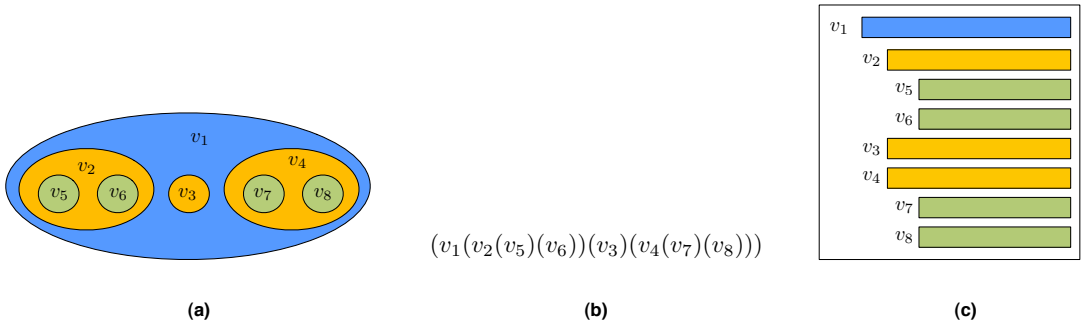


Figure 2.14: Alternative graphical representation of the tree [Knuth, 1997]: (a) nested sets; (b) nested parentheses; (c) indentation.

Figure 2.14a represents an instance of the general idea on *nested sets*: "*A collection of sets in which any pair of sets is either disjoint or one contains the other*" [Knuth, 1997]. This collection of sets represents the structure of the tree mapping every node into a set; the parent-child relationships between the nodes are retained by the inclusion order between the sets. In a tree all the nodes are descendants of the root, in the nested sets all the sets are subsets of the set mapped from the root of the tree.

Figure 2.14b represents a linear nested sets view; indeed, the parentheses can be seen as a set containing other sets. It can be regarded as an algebraic formula involving parentheses.

The parent-child relationships are retained by the nesting inside the parentheses; the principle is exactly the same exploited in the nested sets representation. The representation of the tree in Figure 2.14c works in the same way exploiting the idea of *indentation*: The longest and highest bar contains all the other shorter bars at lower levels; this idea is then applied recursively. The idea of nested sets has been exploited also to define other representations such as the nested intervals, where the sets are represented by intervals with one nested inside the other [Celko, 2000].

Knuth pointed out that: "*The number of different representation methods in itself is ample evidence for the importance of tree structures in every day life as well as in computer programming*" [Knuth, 1997].

Chapter 3

Overview of the Work

In this chapter we present the motivations of the work. Afterwards, we give a preliminary description of the NESTOR Framework composed by the NESTOR Model and the NESTOR Prototype and we detail the main innovations brought about by these components.

3.1 Research Question

We have analyzed the relevance of the concept of hierarchy in many scientific and non-scientific fields highlighting the fundamental role of the tree as the foremost way to represent, model and handle a hierarchy. The role of the tree has gained more and more importance over time thanks to its mathematical formalization based on graph theory which has allowed us to point out many relevant properties, and to define a data model and an abstract data type exploited to model the reality of interest and to design fundamental algorithms in computer science.

From a graphical representation of a hierarchy as we presented the tree – e.g. Figure 2.3b – we moved to a more formal mathematical definition and augmented the possibilities of its use in theoretical and applicative environments. The strong bond between tree and graph theory is underlined even by the usual graphical representation of the tree: a collection of vertices and edges as well as a graph is represented. Indeed, a tree is formally defined as a graph where some conditions of the vertices and edges are respected.

In Figure 2.14 we have seen different graphical representations of the tree. Intuitively, we can see that there is no clear linkage between these representations and graph theory. Instead of vertices and edges there are sets, parentheses and bars (or intervals). The most important representation is the nested sets one, the idea of which is then exploited and adapted to the other representations.

Our main research question is:

> **Are they just graphical representations? Or could we formally define new data models based on these intuitive ideas pointing out new properties?**

As Knuth suggested, the number of different representations of the tree indicate the great impact of this model in computer science and not only. The representation is just the first

step towards a data model that can be adopted to model reality, define properties and design algorithms to solve problems. The idea underlining this work is to take into account the nested sets representation and formalize it in order to design a new data model which can be used as an alternative to the tree. The tree data model has taken great advantage from graph theory, on the one hand exploiting its concepts and properties to deal with hierarchies; on the other hand, the study of trees has given several insights into graph theoretical problems and has helped to address them. Following this pattern the nested sets representation can be related to set theory and thus a set-based data model to deal with hierarchies can be designed and developed in this mathematical field opening up new research possibilities.

As we have seen with the tree the first step is to go from the nested sets representation to a mathematical formalization of a set-based data model. By means of this formalization we can then study the properties of the model and thus put these in relation with the properties and characteristics of the tree model. Once we have pointed out these fundamental characteristics we can model the reality of interest throughout a data model alternative to the tree and study its characteristics in a concrete environment.

After these fundamental steps we have to verify if the expressive power of the newly defined set-based data model can be compared to that of the tree data model. Furthermore, we analyze if *there are any aspects of the reality of interest that we can model by means of the set data model that we could not model with a tree.*

3.2 Contributions of the Work

This work focuses on hierarchical structures and in particular on the tree representation of information. We analyzed the role of hierarchies and trees from a theoretical point-of-view in a way that allows us to relate theoretical concepts to practical issues that arise in the context of Digital Libraries and in particular of the archives and digital archives.

In this work we develop the *NEsted SeTs for Object hieRarchies (NESTOR)* Framework, which envisions an innovative way to represent, manage, access and exchange hierarchies; the Framework is composed by two main components: the NESTOR Model and the NESTOR Prototype.

The NESTOR Model is the heart of the Framework; it is based on two set data models called *Nested Set Model* (NS-M) and *Inverse Nested Set Model* (INS-M) which are based on an organization of nested sets. The foundational idea underlying these set data models is that an opportune set organization can maintain all the features of a tree with the addition of some new relevant functionalities. We define these functionalities in terms of the flexibility of the model, the rapid selection and isolation of easily specified subsets of data and the extraction of only those data necessary for satisfying specific needs. We can use these set data models to represent hierarchical structures by disclosing a variety of properties which can be related to the properties of the tree and which are also peculiar of these models. The representation of hierarchies by one of these models lays the ground for an environment leading to new ways of

modeling and consequently accessing, managing and querying hierarchical data.

The NESTOR Prototype gives an instantiation of the model allowing the application of the formal concepts defined in the NESTOR Model. The prototype is presented by the use case of the archives describing how a hierarchy can be modeled by means of the NESTOR Model and specifically how the archival records can be represented through it. In particular, we face the problem of access and exchange of hierarchically organized resources in distributed environment and discuss the relationships between the NESTOR Prototype and Digital Library technologies such as the Dublin Core metadata, the *Encoded Archival Description (EAD)*, the *Open Archive Initiative - Protocol for Metadata Harvesting* (OAI-PMH) and the *Open Archive Initiative - Object Re-use and Exchange* (OAI-ORE).

3.2.1 The NESTOR Model

The NESTOR Model is the core of the NESTOR Framework because it defines the set data models on which every component of the framework relies. In Figure 3.1 we can see a graphical representation of the principal components composing the NESTOR Model. In the upper part we have the set data models – i.e. the Nested Set Model (NS-M) and the Inverse Nested Set Model (INS-M). The second component represents the properties of the set data models such as: the mapping functions to go from the NS-M to the INS-M and vice versa, the meaning of the union or intersection of two sets and the definition of distance measures. The latter component represents the relationships between the set data models and the tree data model; this component contains the functions for mapping a tree into one of the two models and it compares the properties of the tree with the properties of the set data models.

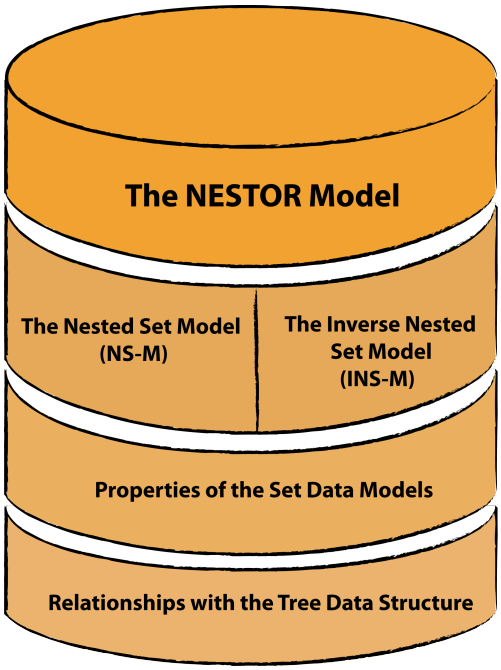


Figure 3.1: The main components of the NESTOR Model.

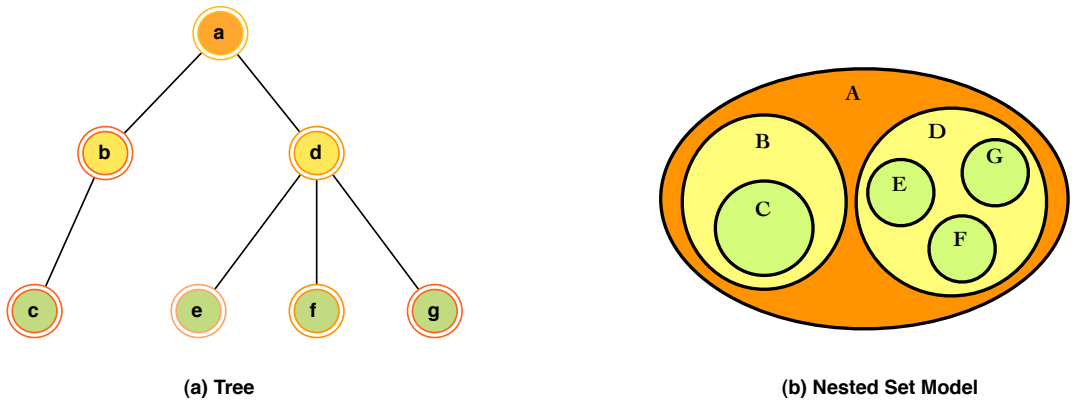


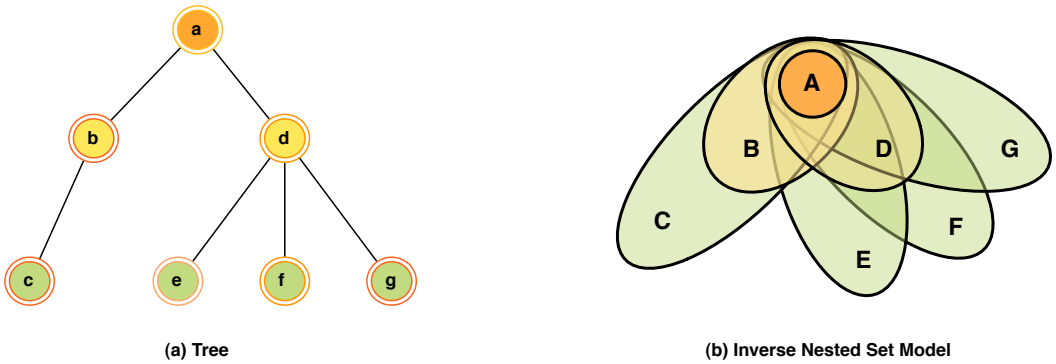
Figure 3.2: (a) A tree. (b) Euler-Venn Diagram of a NS-M

It is worthwhile for describing the state of the art and the contributions of the work to give a brief and intuitive presentation of the ideas underlying the NESTOR Model. The most intuitive way to understand how these models work is to relate them to the tree; thus, we informally present the two data models by means of examples of mapping between them and a sample tree. The first model we introduce is the **Nested Set Model**. An organization of sets in the NS-M is a collection of sets in which any pair of sets is either disjoint or one contains the other. In Figure 3.2b we can see how a sample tree is mapped into an organization of nested sets based on the NS-M.

From Figure 3.2b we can see that each node of the tree is mapped into a set, where child nodes become *proper subsets* of the set created from the parent node. Every set is subset of at least of one set; the set corresponding to the tree root is the only set without any supersets and every set in the hierarchy is subset of the root set. The external nodes are sets with no subsets. The tree structure is maintained thanks to the nested organization and the relationships between the sets are expressed by the set inclusion order. Even the disjunction between two sets brings information; indeed, the disjunction of two sets means that these belong to two different branches of the same tree.

The second data model is the **Inverse Nested Set Model**; in Figure 3.3 we can see how a sample tree is mapped into an organization of nested sets based on the INS-M. We can say that a tree is mapped into the INS-M transforming each node into a set, where each parent node becomes a subset of the sets created from its children. The set created from the tree's root is the only set with no subsets and the root set is a proper subset of all the sets in the hierarchy. The leaves are the sets with no supersets and they are sets containing all the sets created from the nodes composing tree path from a leaf to the root. An important aspect of INS-M is that the intersection of every couple of sets obtained from two nodes is always a set representing a node in the tree. The intersection of all the sets in the INS-M is the set mapped from the root of the tree.

We can see that the intuitive idea on which the NS-M is based comes from the graphical representation of a tree proposed in [Knuth, 1997] to show different ways of representing a tree



(a) Tree (b) Inverse Nested Set Model

Figure 3.3: (a) A tree. (b) Euler-Venn Diagram of an INS-M

– see Section 2.2.4 – but in [Knuth, 1997] there was no definition of a nested set model. This idea was then exploited in [Celko, 2000] to explain an alternative way to solve recursive queries over trees in *Structured Query Language (SQL)* language. The nested sets representation was used to describe an integer intervals encoding of the nodes of a tree; this encoding allows us to query hierarchical data stored in a relational database by exploiting the arithmetic properties of the encoding, avoiding relational joins of the tables and thus avoiding recursion[1]. Although Nested Sets is an elegant technique and certainly appealed to many database developers, it has two fundamental disadvantages [Na and Lee, 2006]: the encoding is volatile and many nodes have to be relabeled whenever a new node is inserted and querying ranges are asymmetric from performance perspective. It is an easy matter to answer if a point falls inside some interval, but it is hard to index a set of intervals that contain a given point. For Nested Sets this translates into a difficulty answering queries about nodes ancestors. This idea has been further developed in the field of databases producing several other hierarchical encodings trying to overcome these issues; for instance, the generalization of the nested sets encoding to the nested intervals one proposed by Tropashko in [Tropashko, 2005] and to encoding schemes that use more sophisticated numbering systems which represent an evolution of both the nested sets view and the classical Dewey encodings [Alkhatib and Scholl, 2009; O'Neil et al., 2004; Tatarinov et al., 2002; Xu et al., 2009].

On the other hand, the idea of the INS-M does not find any reference in the scientific literature. Now we can point out the outcomes of the research activities concerning the set data models composing the NESTOR Model:

- We give *formal definitions based on the set theory* for NS-M and the INS-M; they are both defined as *collections of sets* in which we impose several conditions defining their structures.

[1]The encoding based on integer intervals is often referred to Celko's nested sets. Celko did not claim to be the inventor of this method but just the person that circulated it; indeed, Kamfonas in [Kamfonas, 1992] proposed the same idea to solve recursion in relational databases. The main difference between the two presentations is that Kamfonas did not present the nested sets, but he explained the encoding by using the preorder traversal algorithm on a tree.

- We formally *define and prove the properties of the models*; we define the meaning of set operations on the two set data models determining the result of these operations – e.g. we define which is the outcome of the union of two sets in the NS-M or of the intersection of two sets in the INS-M.

- We define a bijection between the NS-M and the INS-M establishing the mapping functions to go from one model to the other.

- We define a *bijection between both NS-M and INS-M and the tree model*. This definition allows us to establish a formal connection between the tree and the set data models and vice versa. We define a formal correlation between the properties defined in the set data models and in the tree model.

- We define a *distance measure between the sets* in the NS-M and the INS-M called "NS-M graphical distance" and "INS-M graphical distance"; we formally prove the bijection between these distances and the widely-used "graphical distance" on trees [Buneman, 1974; Harary, 1969].

- We define three distance measures between the collection of sets defined in the NS-M or in the INS-M: *Jaccard's distance, the structural distance and the NESTOR distance*. Jaccard's distance defines a distance measure between the content of two collections of sets, without taking into account the structure of the collections; it is an extension to collections of sets of the classical Jaccard's distance [Jaccard, 1901] defined between sets. On the other hand, the structural distance measures the distance between two collections of sets considering only the inclusion order between the sets in the collections. Lastly, the NESTOR distance is a linear combination of these two distances which allows us to weigh the content and the structure components. The three distance measures are proved to be *proper metrics* – i.e. we proved identity, antisymmetry, non-negativity and the triangle inequality.

- We formally prove the NS-M and the INS-M to be *metric spaces*.

The formal definition of the set data models and their properties prove that the nested sets idea is not just an alternative graphical representation of the tree but a proper data model that can be exploited to represent and manage hierarchies. We prove that the expressive power of the set data models and the tree are formally comparable and that the set data models allow us to explicitly represent aspects of the reality that are problematic to catch with the tree. For instance, we can represent a hierarchical organization by means of the sets and then we can represent the objects belonging to the sets and formally establish relations between them. The major concern of the tree model is on the hierarchical structure defined between the entities represented by means of it; the set data models allow us to do the same by means of collections of sets and at the same time to add a further expression dimension represented by the elements belonging to the sets.

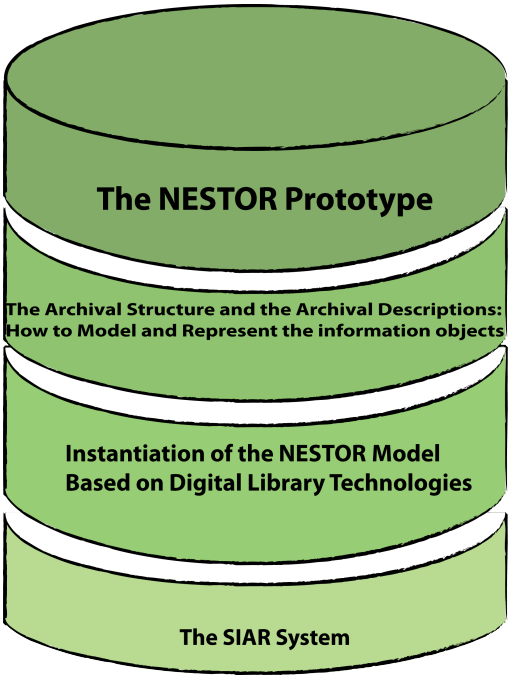


Figure 3.4: The main components of the NESTOR Prototype.

This aspect has a relevant outcome that is a formal separation between the structural and the content aspects of the entities represented within the models. We can deal only with the hierarchical structure of the reality considering the inclusion order between the sets; on the other hand, we can deal only with the content aspects by considering the elements belonging to the sets. These characteristics are highlighted by the definition of two dedicated distance measures, one based on content and the other based on structure. Furthermore, the definition of the NESTOR distance also gives us the possibility of considering both content and structure at the same time. The concrete outcomes outlined by the NESTOR Prototype emphasize the importance of this distinction.

3.2.2 The NESTOR Prototype

The NESTOR Prototype gives an instantiation of the model allowing the application of the formal concepts defined in the NESTOR Model; in Figure 3.4 we can see the main components of the NESTOR Prototype and its implementation in the SIAR system. The first component details how the entities and the information objects we are considering are represented through the NESTOR Model; the second describes the possible instantiations of the model in a concrete environment and the third examines the relationships between the instantiations and the technologies of choice.

The prototype is presented by the use case of the archives describing how a hierarchy can be modeled by means of the NESTOR Model and specifically how the archival records can be represented through it. The NESTOR Prototype takes into account the dimensions of interoperability in Digital Libraries [Gradmann, 2007], showing how the adoption of the NESTOR

Model. In particular, we face the problem of access and exchange of hierarchically organized resources in a distributed environment and discuss the relationships between the NESTOR Prototype and Digital Library technologies. In particular, we analyze the role of EAD, showing how its drawbacks can be overcome by an effective instantiation of the NESTOR Model based on the exploitation of the native functionalities of OAI-PMH and the Dublin Core.

The outcomes of the research activities concerning the NESTOR Prototype are:

- We describe how it is possible to *model an archive and its resources* by means of the NS-M and the INS-M and show that these models let us express some aspects of the reality that are not captured by the tree model.
- We define an *instantiation* of the NESTOR Model which relies on standard Digital Library technologies such as OAI-PMH and the Dublin Core.
- We design a *mapping between the archival description encoded in the standard format EAD and the combination OAI-PMH and Dublin Core* based on the structure defined by the NS-M or the INS-M. We show how this mapping permits us to overcome some well-known issues of the EAD format and at the same time to address interoperability aspects of the archives: variable granularity access and exchange of the archival resources and cross-language access to the archival metadata.
- We show how the NESTOR Model can be used in conjunction with the OAI-ORE data model in order to *represent and manage the aggregation of archival resources* – i.e. defining and dealing with the relationships between archival metadata and full-content digital objects.
- We design and develop *an archival information system based on the NESTOR Model* for the modeling and managing of archival resources; we show how this choice allows us to separate the management of structural and content aspects of the archives. This archival information system takes into account all the aspects of management and curation of archival resources from the format and the encoding of archival metadata to their exchange in a distributed environment. We take care of the users developing different user interfaces and conducting user studies.

Chapter 4

Mathematical Background

In this chapter we present the mathematical concepts we are going to exploit in the rest of the work. First-of-all we present the basics of set-theory. In the second part of the chapter we describe the main concepts of graph theory; in particular, we focus on the mathematical definition and properties of the tree.

4.1 Set Theory

4.1.1 A Historical Glance

Among all mathematical disciplines, set theory occupies a special place because it plays two different roles at the same time: on the one hand, it is an area of mathematics devoted to the study of abstract sets and their properties; on the other, it provides mathematics with its foundation.

Set theory began with the work of Cantor that from 1874 to 1884 published a series of works constituting the origin of the field. Among all the meaningful results presented by Cantor, we focus on three main concepts that strongly influenced the history of set theory: *countable and uncountable infinite sets*, *ordinals*, and *well-ordered sets*.

The first concept was introduced by Cantor in 1874 when he proved a fundamental result showing that infinite sets can be of different sizes – i.e. he proved that there are more real numbers than algebraic ones. What does it means that there are more real numbers than natural ones, when they are both infinite? Cantor defined two sets, say A and B, to have the same size, or *cardinality*, if there is a bijection between them: that is, if there is a one-to-one correspondence between the elements of A and the elements of B. If there is no bijection between these two sets, but there is a bijection between A and a subset of B, then A is of a *smaller cardinality* than B. Following this reasoning Cantor proved the existence of countable and uncountable infinite sets showing that - as an example – the set of naturals ($\mathbb{N}$) is a countable infinite and the set of reals ($\mathbb{R}$) is an uncountable infinite.

The second and the third concepts are: *ordinals* and *well-ordered set*. Both of them settled the ground for a fundamental development in set theory: the *Zermelo-Fraenkel Set Theory*.

We can describe the *ordinals* by using the natural numbers to count a collection of objects, we assign a number to each object, starting with 1, continuing with 2,3, etc. and stopping when we have counted each object exactly once. When this process is over we have obtained a number n, the last number in the sequence that tells us how many objects there are in the collection. We have also defined an *ordering* on the objects that we were counting, namely the order in which we count them. In the first case we think about a set in terms of *size*; then if we have a set X in a one-to-one correspondence with $\{1, 2, \ldots, n\}$, we conclude that X has cardinality n. But we can also take note of the natural ordering on the set $\{1, 2, \ldots, n\}$, in which case we observe that our one-to-one correspondence provides us with an ordering on X too. If we adopt the first point-of-view we are regarding n as a *cardinal*, and if we adopt the second we are regarding it as an *ordinal*.

In set theory, one likes to regard all the objects as sets; if we consider an ordinal n, it is identified with the set of all its predecessors [Gowers et al., 2008]; for instance, the ordinal n is identified with the set $\{1, 2, \ldots, n\}$. Each ordinal is a *well-ordered set*, which means that every non-empty subset of it has a least element. Cantor defined the *well-ordered principle*: the assertion that every set can be well-ordered; this principle led to some paradoxes outlined by Burali-Forti and Russell [Potter, 2004] that put at risk further developments of the set theory.

The Russell paradox starts from the intuitive notion that every property determines a set – i.e. the set of those objects that have that property. If we consider the property of *being an ordinal number*, we determine the set of all ordinals. We can see that a set like that cannot exist since it would be *well-ordered* and would therefore correspond to an ordinal greater than all ordinals, which is an absurd. Similarly, the property of *being a set that is not an element of itself* cannot determine a set because if A is such a set, then A is an element of A if and only if A is not an element of A, which is impossible. The lesson contained in Russell's paradox is that it is not enough to define a set via a property to prove its existence.

So, the problem was: what is a set? The task for set-theorists was to determine the properties which defined a set; the idea that followed was to establish a set of axioms, and take care to check that all the theorems follow logically from the axioms. Zermelo in 1908 tried to capture this idea of set in a short list of basic principles later improved by Fraenkel, Skolem and Von Neumann, becoming what is now known as the *Zermelo-Fraenkel Set Theory with the Axiom of Choice* or ZFC. By means of the ZFC, Zermelo gave a proof of the well-ordering principle [Moore, 1982]. The basic idea behind the axioms of ZFC is that there is a "*universe of all sets*" that we would like to understand, and the axioms give us the tools we need to build sets out of other sets. The ZFC axioms are: Extensionality, Power Set, Infinity, Replacement, Union, Regularity, and Choice; a complete and detailed treatment of the ZFC can be found in [Jech, 2003]. The axioms of ZFC are "*generally accepted as a correct formalization of those principles that mathematicians apply when dealing with sets*" [Jech, 2003].

ZFC has been fundamental in the twentieth century to prove some important results as complements to Cantor's work: the proof of the consistency of continuum hypothesis[1] by [Gödel,

[1]The continuum hypothesis was proposed by Cantor in 1877: *There is no infinite set with a cardinal number be-*

1940] and the independence of the continuum hypothesis by Coen [Cohen, 1963] which established the impossibility to prove the hypothesis [Gowers et al., 2008]. Furthermore, "*experience has shown that practically all notions used in contemporary mathematics can be defined, and their mathematical properties derived, in this axiomatic system*" [Hrbacek and Jech, 1999]. In this sense the ZFC serves as a satisfactory foundation for the other branches of mathematics.

The ZFC set theory is a very powerful and quite complex axiomatic system and in our work we are going to exploit only some of its axioms from which we derived some useful definitions and theorems.

4.1.2 Sets, Properties and Operations

Many sentences of mathematics can be rewritten in set-theoretic terms and often there are circumstances where it becomes extremely convenient. For examples, one of the great advances in mathematics was the use of Cartesian coordinates to translate geometry into algebra and the way this was done was to define geometrical objects as sets of points, where points were themselves defined as pairs or triples of numbers [Gowers et al., 2008].

A second circumstance where it is usually hard to do without sets is when one is defining new mathematical objects. Very often such an object is a set together with a mathematical structure imposed on it, which takes the form of certain relationships among the elements of the set [Davey and Priestley, 2002].

Broadly speaking a set is a collection of objects, and in mathematical discourse these objects are mathematical ones such as numbers, points in space, or even other sets. In a broader sense, a set is a collection of distinguishable objects, called its elements; the assumptions underlying classical set theory is spelt out by [Kamke, 1950]: "By a set we are to understand, according to G. Cantor, *a collection into a whole, of definite, well-distinguished objects (called the elements) of our perception or of our thought [...]. For a set, the order of succession of its elements shall not matter [...]; furthermore, the same element shall not be allowed to appear more than once.*".

If an object a is an element of A we write $a \in A$, otherwise we write $a \notin A$. There are three common ways to denote a specific set. One is to list its elements inside curly brackets. For instance, we can define a set A containing the elements a, r, c by writing $A = \{a, r, c\}$. If no further specifications are provided, we refer to the basic definition of set where elements are not ordered and they cannot appear more than once. We indicate the elements belonging to a set with lower cases, i.e. $A = \{a, c, f, b\}$ means that the set A is composed by the elements a, b, c, f (we can see that the order between the letters has no meaning). Occasionally, we may indicate the elements belonging to a set, say X as $x_1, x_2, \ldots, x_n$; even in this case if it is not clearly stated, the elements are not considered to be ordered in the set X.

The majority of sets considered by mathematicians are too large for this to be feasible [Gowers et al., 2008] – indeed they are often infinite – so a second way to denote sets is to use

tween that of the "small" infinite set of integers $\aleph_0$ *and the "large" infinite set of real numbers c (the "continuum"). Symbolically, the continuum hypothesis is that* $\aleph_1 = c$ [Jech, 2003].

dots to imply a list that is to long to write down: for instance, $A = \{1, 2, 3, \ldots, 100\}$ and $A = \{1, 2, 3, \ldots\}$.

A third way, that is the most important [Gowers et al., 2008], is to define a set via a *property*: for instance, we can define the set A of even integers as $A = \{x : x \text{ is prime} \wedge x > 10\}$. The colon in this notation is read "such that"; in this instance, we read the definition of set A as: "The set of x such that x is a prime *and* x is greater than 10." It is also possible to use the vertical bar "|" in place of the colon: $A = \{x \mid x \text{ is prime} \wedge x > 10\}$.

We adopt a special notation for frequently encountered sets such as $\emptyset$ denotes the *empty set*, that is, the set containing no elements, $\mathbb{N}$ which denotes the set of *natural numbers*, $\mathbb{Z}$ which denotes the set of *integers*, and $\mathbb{R}$ which denotes the set of *real numbers*.

It is important to highlight that all the set-theoretic properties can be stated in terms of membership with the help of logical means: identity, logical connectives and quantifiers. An obvious fact about *identity* is: $A = B$ then $B = A$, that means A is identical to B then B is identical to A. Two sets are *identical* if they contain the same elements, written $A = B$. *Logical connectives* can be used to construct more complicated properties from simpler ones; they are expressions like "not" ($\neg$), "and" ($\wedge$), "or" ($\vee$), "then" ($\Rightarrow$), "if and only if" ($\Leftrightarrow$). *Quantifiers* like "there exists" ($\exists$) "there exists and it is unique" $\exists!$, and "for all" ($\forall$) provide additional logical means. Let us see an example of property that involves some of these logical means: $\forall X \in Y, \exists Z \mid Z \in X \wedge Z \in Y$ that is read "*for all* X belonging to Y, *there exists* Z *such that* Z belongs to X *and* Z belongs to Y".

The definition of the fundamental set-theoretical operations (i.e. union, intersection and difference) are based on the elements belonging to the sets. Indeed, let A and B be two sets, then:

Union. $A \cup B = \{x : x \in A \vee x \in B\}$

Intersection. $A \cap B = \{x : x \in A \wedge x \in B\}$

Difference. $A \setminus B = \{x : x \in A \wedge x \notin B\}$

Symmetric Difference. $A \triangle B = (A \setminus B) \cup (B \setminus A)$

The symmetric difference between two sets A and B can also be defined as the union of A and B minus their intersection and it is written $A \triangle B = (A \cup B) \setminus (A \cap B)$. Two sets A and B are *disjoint* if they have no elements in common – i.e. $A \cap B = \emptyset$.

As we have said all the set-theoretical properties can be stated by means of membership properties and logical means but with more complicated definitions and theorems it is practical to give names to some widely-used properties. If all the elements of a set A are contained in a set B, that is, if $a \in A$ implies $a \in B$, then we write $A \subseteq B$ and say that A is a *subset* of B.

Subset. $A \subseteq B = \{a : a \in A \Rightarrow a \in B\}$

A set A is a *proper subset* of B, written $A \subset B$, if $A \subseteq B$ but $A \neq B$.

4.1.3 Relations and Functions

In this section we see how various mathematical concepts like relations and functions can be expressed by means of sets. We begin by introducing the concept of *ordered pair*. If a and b are elements then the unordered pair $\{a, b\}$ is the set whose elements are exactly a and b. The order plays no role and thus, $\{a, b\} = \{b, a\}$. In many applications we need to indicate which element in a pair comes first and thus, we have to define an order between the elements of the pair. We indicate with (a, b) an *ordered pair* where a is the first coordinate and b is the second coordinate. An ordered pair has to be a set and it can be defined in such a way that $(a, b) = (a', b')$ if and only if $a = a'$ and $b = b'$. With ordered pairs at our disposal we can define ordered triples $(a, b, c) = ((a, b), c)$ and ordered quadruples $(a, b, c, d) = ((a, b, c), d)$ and so on and so forth.

From ordered pairs it follows the central concept of *binary relation*. A binary relation is determined by specifying all the ordered pairs of objects in that relation.

Definition 4.1.
*A set R is a **binary relation** if all elements of R are ordered pairs, i.e. let A and B be two sets, if for any $r \in R$ there exists $a \in A$ and $b \in B$ such that $r = (a, b)$.*

Example 4.1. *Let us consider the relation $R_2 = \{r : \exists a, b \in \mathbb{Z} \mid r = (a, b) \wedge b \setminus a \in \mathbb{Z}\}$ meaning that R_2 is a relation composed by all those ordered pairs $r = (a, b)$ such that $b \setminus a$ is an integer. The elements of R_2 are the ordered pairs:* $(1, 1), (1, 2), (1, 3), (1, 4), \ldots, (2, 2), (2, 4), (2, 6), \ldots, (3, 3), (3, 6), (3, 9), \ldots$, *and so on and so forth.*

It is customary to write aRb in the place of $(a, b) \in R$; so, a is in relation R with b if aRb holds. Let us consider the relation aRb, we call *domain of the relation* the set of all a which are in relation R with some b and it is denoted with $\mathrm{dom}(R)$. The set of all b such that, for some a, a is in relation R with b is called the *range of the relation* and it is indicated with $\mathrm{ran}(R)$. We can say that $\mathrm{dom}(R)$ is the set of all the first coordinates of R and $\mathrm{ran}(R)$ is the set of all the second coordinates of R.

Definition 4.2.
*Let A be a set and R a relation, then we define $R[A]$ to be the **image** of A under R where $R[A] = \{b : b \in ran(R) \wedge \exists a \in A \mid aRb\}$.*

The image of A under R is the set of all b from the range of R related in R to some element of A.

Example 4.2. *Let us consider the relation R_2 of Example 4.1: $R_2 = \{r : \exists a, b \in \mathbb{Z} \mid r = (a, b) \wedge b \setminus a \in \mathbb{Z}\}$. Then, let us consider the set $A = \{3, 8, 9, 12\}$ and $B = \{2\}$.*

Definition 4.3.
Let R and S be binary relations. The composition of R and S is the relation:

$$T = S \circ R = \{aTc \mid \exists b \text{ for which } aRb \wedge bSc\}.$$

So $(a, c) \in S \circ R$ means that for some b, aRb and bSc. To find objects related to a in $S \circ R$, we first find objects b related to a in R, and then objects related to those b in S. Please note that R is performed first and S second but notation $S \circ R$ is customary [Hrbacek and Jech, 1999].

A very important relation is the *cartesian product* between two sets. The set of all ordered pairs whose first coordinate is from A and whose second coordinate is from B is called the Cartesian product of A and B and denoted by $A \times B$.

Definition 4.4.
Let A and B be sets, then the ***Cartesian product*** *is defined to be:*

$$A \times B = \{(a, b) \mid a \in A \wedge b \in B)\}.$$

Thus $A \times B$ is a relation in which every element of A is related to every element of B. Notice that $A \times B \times C = \{(a, b, c) \mid a \in A \wedge b \in B \wedge c \in C\}$.

The concept of relation is strictly related to the concept of function; indeed, a function is a procedure assigning to any object a from the domain of the function a unique object b in the range, namely, to the value of the function at a.

Definition 4.5.
A binary relation F is called a function (or mapping) if aFb_1 and aFb_2 imply $b_1 = b_2$ for any a, b_1 and b_2. In other words, a binary relation F is a function if and only if for every a from $\mathrm{dom}(F)$ *there is exactly one b such that aFb. This unique b is called the value of F at a and is denoted by $F(a)$.*

In the rest of the work a function is not necessarily indicated with caps – e.g. F – but also with small caps – e.g. f – and Greek letters – e.g. φ or ϕ. If F is a function with $\mathrm{dom}(F) = A$ and $\mathrm{ran}(F) \subseteq B$, it is customary to write $F : A \to B$. The range of the function can be denoted $\{F(a) \mid a \in A\}$ or $\{F_a\}_{a \in A}$. It is worthwhile for the rest of the work to point out some important definitions related to the concept of function.

Definition 4.6.
Let F be a function and A, B and C be sets such that $C \subseteq A$.

$$F \text{ is a function on } A \text{ if } \mathrm{dom}(F) = A. \tag{4.1}$$

$$F \text{ is a function into } B \text{ if } \mathrm{ran}(F) \subseteq B. \tag{4.2}$$

$$F \text{ is a function onto } B \text{ if } \mathrm{ran}(F) = B. \tag{4.3}$$

$$\text{The restriction of the function } F \text{ to } C \text{ is the function } F|_C = \{(a, b) \in F \mid a \in C\}. \tag{4.4}$$

The last important definition about functions regards the concepts of *injective*, *surjective* and *bijective* functions.

Definition 4.7.
Let A and B be two sets and $F : A \to B$ a function, then:

Injection. *F is an injective function if:* $\forall a_1, a_2 \in A, \exists b_1, b_2 \in B, b_1 \neq b_2 \mid b_1 = F(a_1) \wedge F(a_2) = b_2$.

Surjection. *F is a surjective function if:* $\forall b \in B, \exists a \in A \mid F(a) = b$.

Bijection. *F is a bijective function if it is both injective and surjective.*

A consequence of the definition of injective function is that if $F : A \to B$ is injective and there exists $a_1, a_2 \in A$ such that $F(a_1) = b$ and $F(a_2) = b$ then implies that $a_1 = a_2$. If $F : A \to B$ is a surjective function then $ran(F) = B$; if it is a bijective function, then $\mathrm{dom}(F) = A$ and $ran(F) = B$: each element of A corresponds to a unique element of B and vice versa.

4.1.4 Collections and Families of Sets

The concept of *collection of sets* or *system of sets* derives from the **Axiom of Union** of the ZFC:

> *For any set C there exists a set U such that $x \in U$ if and only if $x \in A$ for some $A \in C$.*

U is called the union of C and it is indicated with $\bigcup C$; we say that C is a collection of sets when we want to stress the fact that the elements of C are sets – we use the notation C to indicate a collection of sets. We denote a collection of sets C by means of the same ways for denoting the sets: list its elements – e.g. $C = \{A, D, E, T\}$ where A, D, E, T are sets – and via a property – e.g. $C = \{A \subset \mathbb{N} \mid |A| < 3\}$.

We can extend the concepts described for sets also to collection of sets. If C is a collection of sets, then x is an element of the *union* of C if and only if for at least one element A of C, x is an element of A: $x \in \bigcup C \Leftrightarrow \exists A \in C \mid x \in A$. The notation can vary: $\bigcup C$ or $\bigcup_{A \in C} A$. The specular

definition and notation is adopted for the intersection; x is an element of the *intersection* of C if and only if for every set A in C, x is an element of A: $x \in \bigcap C \Leftrightarrow \forall A \in C, x \in A$. It is worthwhile to point out the differences between the *cardinalities* of a set and of a collection of sets; we call $|A|$ the cardinality of a set A and it is the number of elements belonging to A. We call $|C|$ the cardinality of a collection of sets C and it is the number of sets belonging to C; the total number of elements belonging to the collection C is $|\bigcup C|$.

An important collection of sets is derived from the **Axiom of Power Set** of the ZFC:

For any set A there exists a set P such that $X \in P$ if and only if $X \subseteq A$.

Since P is uniquely determined, we call the set of all the subsets of A the *power set* of A and indicate it by $\mathcal{P}(A)$. For instance, $\mathcal{P}(\emptyset) = \{\emptyset\}$, and $\mathcal{P}(a) = \{\emptyset, \{a\}\}$; so, $|\mathcal{P}(A)| = 2^{|A|}$.

A fundamental concept for the rest of the work is the one of *family of sets* or alternatively called *indexed collection of sets*. Please note that the terms family of sets and family of subsets are used as synonyms in the literature.

Definition 4.8.
*Let A be a set, I a non-empty set and C a collection of sets of A. Then a bijective function $\mathcal{A} : I \to C$ is a **family** of sets of A. We call I the **index** set and we say that the collection C is **indexed** by I.*

It is possible to use the extended notation $\{A_i\}_{i \in I}$ to indicate the family $\mathcal{A}$; it is also possible to indicate a family of sets of A as: $\langle A_i \mid i \in I \rangle$. The notation $A_i \in \{A_i\}_{i \in I}$ means that $\exists\, i \in I \mid \mathcal{A}(i) = A_i$. In the rest of the work to indicate a family of sets we will use the shorthand notation $\mathcal{A}_I$ which means that we are considering the collection of sets $\mathcal{A}$ indexed by the set I; as we can see the shorthand notation stresses the fact that a family is an indexed collection. Given that the all the work that follows is based on the concepts of collection and family of sets it is worthwhile to summarize the notation that will be used:

Collection of sets. We indicate a collection of sets as C.

Family of sets. We indicate a family of sets as $\mathcal{A}_I$ to say that the collection $\mathcal{A}$ is indexed by the index set I.

The concepts defined for collection of sets are straightforwardly extended to the families of sets. For instance, $\bigcup \mathcal{A}_I$ indicates the union of all the sets in the family $\mathcal{A}_I$ – please note that the notation $\bigcup_{i \in I} A_i$ is also extensively used.

As we have done with the most used properties of sets, it is practical to give names to some widely-used families of sets.

A frequently used concept is the one of **subfamily**: We indicate with $\mathcal{A}_J$ the **subfamily** of $\mathcal{A}_I$ defined as its **restriction** to $J \subseteq I$ and we say that $\mathcal{A}_J \subseteq \mathcal{A}_I$. Another useful concept is the concept of linearly ordered family or chain.

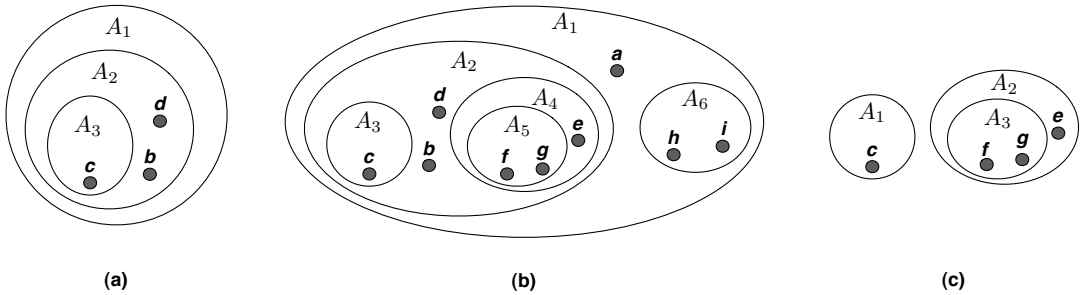


Figure 4.1: (a) A topped and linearly ordered family. (b) A topped but not linearly ordered family. (c) A non-topped family.

Definition 4.9.
Let $\mathcal{A}_I$ be a family. If $\forall A_j, A_k \in \mathcal{A}_I, A_j \subseteq A_k \vee A_k \subseteq A_j$ then $\mathcal{A}_I$ is defined to be a ***linearly ordered family****.*

It is also useful to introduce the concept of non-comparability between two sets [Davey and Priestley, 2002].

Definition 4.10.
Let $\mathcal{A}_I$ be a family of sets and $A_j, A_k \in \mathcal{A}_I$ be two sets. If $A_j \nsubseteq A_k \wedge A_k \nsubseteq A_j$ then A_j and A_k are defined to be ***incomparable*** *or non-comparable, denoted with $A_j \| A_k$.*

In the following we will frequently use the concept of topped family of sets.

Definition 4.11.
Let $\mathcal{A}_I$ be a family. We define $\mathcal{A}_I$ to be a ***topped family*** *if $\exists A_k \in \mathcal{A}_I \mid \forall A_j \in \mathcal{A}_I, A_j \subseteq A_k$.*

From this definition it follows a proposition showing that every linearly ordered family must be topped.

Proposition 4.1. *Let A_I be a linearly ordered family. Then, A_I is a topped family.*

Proof:
We prove this proposition by induction on the cardinality of $\mathcal{A}_I$.

Base: $|\mathcal{A}_I| = 1$. In this case the family $\mathcal{A}_I = \{A_1\}$ is composed by one set thus, it is linearly ordered and topped.

Hypothesis: $|\mathcal{A}_I| = n$. We assume that $\mathcal{A}_I = \{A_1, \ldots, A_n\}$ is linearly ordered and topped, and without any loss of generality we assume that A_1 is the common superset (top set) of all the sets in the family.

Inductive Step: $|\mathcal{A}_I| = n + 1$. *We know that* $\mathcal{A}_I = \{A_1, \ldots, A_n\}$ *is linearly ordered and topped by* A_1. *Ab absurdo suppose that* $\mathcal{A}_I \cup A_{n+1}$ *is linearly ordered but not topped; this means that* $A_{n+1} \not\subseteq A_1 \wedge A_1 \not\subseteq A_{n+1} \Rightarrow \mathcal{A}_I \cup A_{n+1}$ *is not linearly ordered.* □

In Figure 4.1 we can see three families of sets, where the first is both a linearly ordered family and a topped family, the second is topped but not linearly ordered and the third is a non-topped family, where we can see that there is no common set containing all the sets in the family. The following definitions point out two main concepts that we are going to exploit extensively in this work which are the *collection of proper subsets/supersets* and the *collection of direct subsets/supersets*.

Definition 4.12.
Let $\mathcal{A}_I$ *be a family and* $A_j \in \mathcal{A}_I$ *be a set. We define* $\mathcal{S}^+(A_j) = \{A_k \in \mathcal{A}_I \mid A_k \subset A_j\}$ *to be the* ***collection of proper subsets*** *of* A_j *in the family* $\mathcal{A}_I$.
We define the ***collection of proper supersets*** *of* A_j *in the family* $\mathcal{A}$ *as* $\mathcal{S}^-(A_j) = \{A_k \in \mathcal{A}_I \mid A_j \subset A_k\}$.

Please note that $\mathcal{S}^+(A_j)$ is a collection of subsets related to the family of sets $\mathcal{A}$, if there is any risk of ambiguity we add a subscript to the notation to highlight this fact: $\mathcal{S}^+_{\mathcal{A}}(A_j)$. We shall do the same for the collection of supersets: $\mathcal{S}^-_{\mathcal{A}}(A_j)$.

If we consider the collection of subsets of $A_j \in \mathcal{A}_I$, that is $\mathcal{S}^-(A_j)$, we take into account the definition of family $\mathcal{A} : I \to C$, then we can say that $\mathcal{S}^-(A_j)$ is a subset of the $ran(\mathcal{A})$ defined as the collection of all the sets in $\mathcal{A}_I$ which are also *proper* subsets of $A_j \in \mathcal{A}_I$. The meaning is symmetric for the collection of supersets. It is worthwhile for the rest of the work to introduce the definition of *collection of direct super/sub-sets* as a restriction of the collection of proper super/sub-sets.

Definition 4.13.
Let $\mathcal{A}_I$ *be a family and* $A_j \in \mathcal{A}_I$ *be a set. We define* $\mathcal{D}^+(A_j) = \{A_k \in \mathcal{A}_I \mid (A_k \subset A_j) \wedge (\nexists A_t \in \mathcal{A}_I \mid A_k \subset A_t \subset A_j)\}$ *to be the* ***collection of direct subsets*** *of* A_j *in the family* $\mathcal{A}_I$.
We define the ***collection of direct supersets*** *of* A_j *in the family* $\mathcal{A}_I$ *as* $\mathcal{D}^-(A_j) = \{A_k \in \mathcal{A}_I \mid (A_j \subset A_k) \wedge (\nexists A_t \in \mathcal{A}_I \mid A_j \subset A_t \subset A_k)\}$.

The following example explains the meaning of proper collection of subsets/supersets and it shows how we can use the general notion of union of a collection of sets.

Example 4.3. *Let* $\mathcal{A}_I$ *be a family and let* $A_1, A_2, A_3, A_4, A_5, A_6 \in \mathcal{A}_I$ *where* $A_1 = \{a, b, c, d, e, f, g, h, i\}$, $A_2 = \{b, d\}$, $A_3 = \{c\}$, $A_4 = \{e\}$, $A_5 = \{f, g\}$, *and* $A_6 = \{h, i\}$. *In Figure 4.2 we can see a graphical representation by means of Euler-Venn diagrams of this family.*

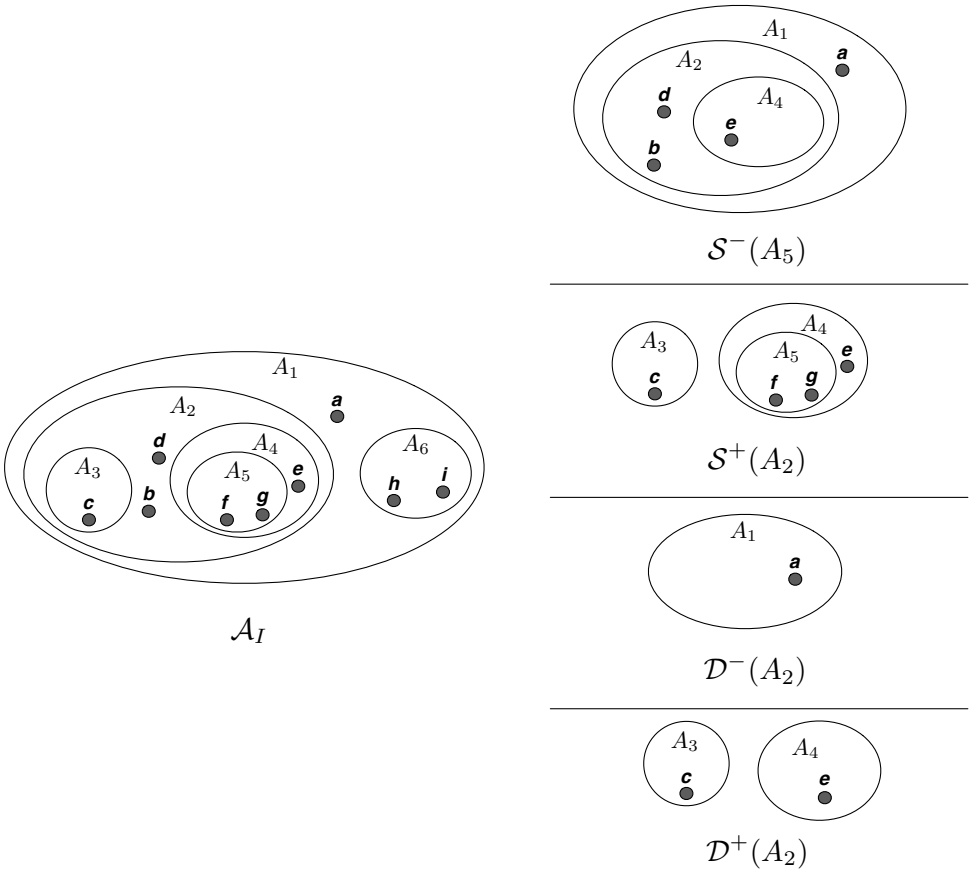


Figure 4.2: The family $\mathcal{A}_I$ (see Figure 4.1b) and a several collections of sets defined in Example 4.3.

From Definition 4.12, it follows that: $\mathcal{S}^-(A_1) = \emptyset$, $\mathcal{S}^+(A_1) = \{A_2, A_3, A_4, A_5, A_6\}$, $\mathcal{S}^-(A_2) = \{A_1\}$, $\mathcal{S}^+(A_2) = \{A_3, A_4, A_5\}$, *and so on and so forth.*

Moreover $\bigcup \mathcal{S}^-(A_1) = \emptyset$, $\bigcup \mathcal{S}^+(A_1)) = \{a, b, c, d, e, f, g, h, i\}$, $\bigcup \mathcal{S}^-(A_2) = \{a\}$, $\mathcal{S}^+(A_2) = \{b, c, d, e, f, g\}$, *and so on and so forth.*

From Definition 4.13 follows that: $\mathcal{D}^-(A_1) = \emptyset$, $\mathcal{D}^+(A_1) = \{A_2, A_3\}$, $\mathcal{D}^-(A_2) = \{A_1\}$, $\mathcal{D}^+(A_2) = \{A_3, A_4\}$ *(as we can see in Figure 4.2), and so on and so forth.*

An important concept is the order-isomorphism between two families.

Definition 4.14.
Let $\mathcal{A}_I$ *and* $\mathcal{B}_J$ *be two families, and* $\varphi : \mathcal{A}_I \to \mathcal{B}_J$ *be a bijective function. We define* $\mathcal{A}_I$ *and* $\mathcal{B}_J$ *to be* ***order-isomorphic*** *if:* $\forall A_k, A_t \in \mathcal{A}_I, A_k \subset A_t \Leftrightarrow \varphi(A_k) \subset \varphi(A_t)$.

The function φ faithfully mirrors the inclusion order structure of the family $\mathcal{A}_I$ in the family $\mathcal{B}_J$ and it is necessarily **bijective**. A related concept is order-embedding that constitutes a notion strictly weaker than the concept of an order-isomorphism.

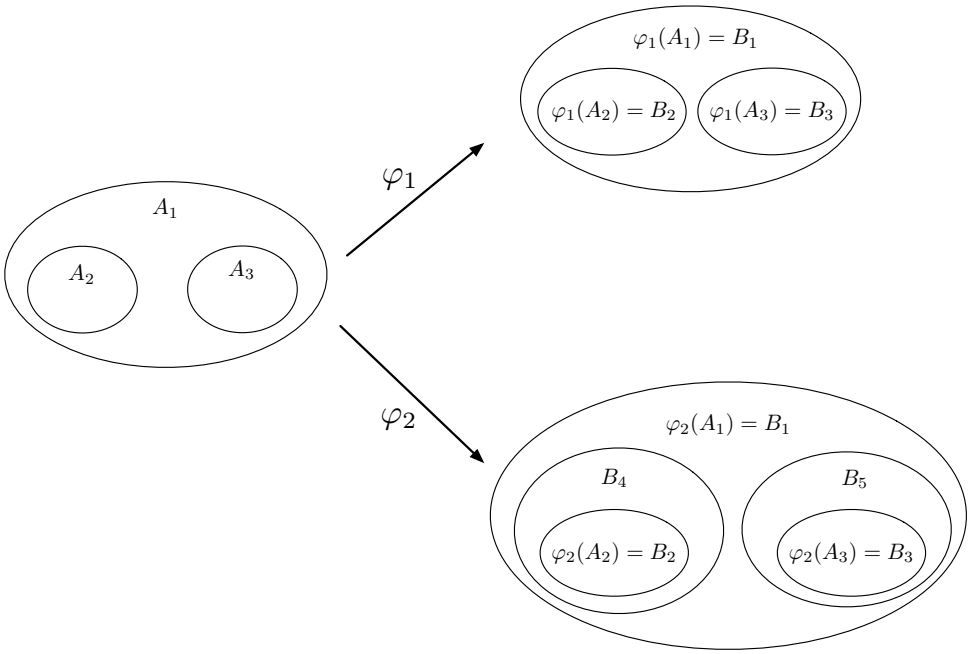


Figure 4.3: An example of order-isomorphism (φ_1) and an example of order-embedding (φ_2).

Definition 4.15.
Let $\mathcal{A}_I$ and $\mathcal{B}_J$ be two families, and $\varphi : \mathcal{A}_I \to \mathcal{B}_J$ be an injective function. We define φ to be an ***order-embedding*** *if: $\forall A_k, A_t \in \mathcal{A}_I, A_k \subset A_t \Leftrightarrow \varphi(A_k) \subset \varphi(A_t)$.*

In this case we say that the family $\mathcal{B}_J$ is an embedding of the family $\mathcal{A}_I$. An order-isomorphism is also an order-embedding but the vice versa is not necessarily true; the concepts are related as well as the concepts of injection and bijection: a bijective function is also injective but the vice versa is not necessarily true. It is important to appreciate the difference between the notion of order-embedding and order-isomorphism; in Figure 4.3, φ_2 is an order-embedding but not an order-isomorphism, instead φ_1 is both an order-embedding and an order-isomorphism. In this context we take into account only the order inclusion between sets without considering the elements that they contain, for this reason we do not draw any elements in the sets.

4.1.5 Ordered Sets

Order is not a property intrinsic to a single object. It concerns comparison between pairs of objects. This section focuses on the concept of ordered set, which is a set equipped with a special type of binary relation.

Definition 4.16.
Let A be a set. An ***order*** *(or* ***partial order****) on A is a binary relation $\leq$ on A such that,*

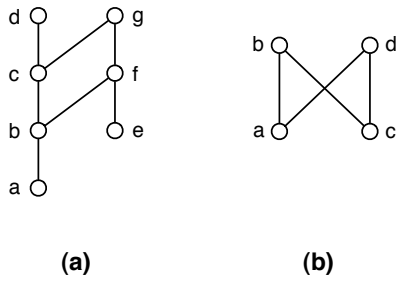


Figure 4.4: Two Hasse diagrams representing two posets.

$\forall x, y, z \in A$:

$$x \leq x \tag{4.5}$$

$$x \leq y \wedge y \leq x \Rightarrow x = y \tag{4.6}$$

$$x \leq y \wedge y \leq z \Rightarrow x \leq z \tag{4.7}$$

A set A equipped with an order relation $\leq$ is said to be an ordered set (or partially ordered set); we use the shorthand name **poset**. When it is necessary to point out the order relation we write $\langle A, \leq \rangle$. An order relation $\leq$ on a set A gives rise to the relation $<$ of strict inequality: $x < y$ in A if and only if $x \leq y$ and $x \neq y$. It is possible to restate the conditions 4.5–4.7 above in terms of $<$, and so to regard $<$ as the fundamental relation [Davey and Priestley, 2002]. In the previous Section we have showed the use of the symbol $\|$ which indicates *non-comparability* between two sets in a collection; we can extend the use of $\|$ to the elements of a poset – i.e. we write $x \| y$ if $x \not\leq y \wedge y \not\leq x$.

We can express the structure of the items using the so called **Hasse diagrams** [Davey and Priestley, 2002]. In order to define an Hasse diagram, we need to define the concept of covering relation between the items belonging to a set.

Definition 4.17.
*Let A be an ordered set and let $x, y \in A$, we say that y is **covered** by x, $y \prec x$ if and only if $y < x \wedge \nexists k \in A \mid j < k < x$.*

Now we can define a Hasse diagram for an ordered set following the construction defined in [Davey and Priestley, 2002].

Definition 4.18.
*Let A be an ordered set, we define with $H(A)$ its **Hasse diagram**.*

A Hasse diagram represents an ordered set by a configuration of circles and interconnecting lines indicating the covering relations. The construction of a Hasse diagram goes as follows:

1. *To each set $i \in A$ associate a point $p(i)$ of the Euclidean plane $\mathbb{R}^2$ depicted by a small*

circle.

2. *For each covering pair $j \prec i$ in A take a line segment $l(i, j)$ joining the circle $p(i)$ to $p(j)$.*

3. *Carry out (1) and (2) in such a way that*

   (a) *if $(j \prec i)$ then $p(j)$ is lower than $p(i)$.*

   (b) *For all $k \in A, k \neq \{i, j\}$, $p(k)$ does not intersect $l(i, j)$ if $k \neq i \wedge k \neq j$.*

It is easy to tell form a diagram whether one element of an ordered set is less than another: *$x < y$ if and only if there is a sequence of connected line segments moving upwards* [Davey and Priestley, 2002] from x to y.

Example 4.4. *Let us consider a poset $\langle A, < \rangle = \{a, b, c, d, e, f, g\}$ where $a < b < c < d$, $e < f < g$, $c < g$, and $b < f$. We can see the Hasse diagram of this poset in Figure 4.4a. From this diagram it is easy to tell that $a < g$ and that $e \| a$.*

Let us consider the poset represented in Figure 4.4b. In this case we can see that $a \| c$ and $b \| d$. Thanks to the Hasse diagram it is quite straightforward to see if two elements are non-comparable.

The diagrammatic approach to finite ordered sets is made fully legitimate by Proposition 4.3 which follows from Lemma 4.2 [Davey and Priestley, 2002]. The following lemma relies on Definition 4.22.

Lemma 4.2. *Let P and Q be finite ordered sets and let $\varphi : P \to Q$ be a bijective function. Then the following are equivalent:*

$$\varphi \text{ is an order-isomorphism.} \tag{4.8}$$

$$x < y \text{ in } P \text{ if and only if } \varphi(x) < \varphi(y) \text{ in } Q. \tag{4.9}$$

$$x \prec y \text{ in } P \text{ if and only if } \varphi(x) \prec \varphi(y) \text{ in } Q. \tag{4.10}$$

Proof:
The proof of this Lemma can be found at page 13 of [Davey and Priestley, 2002].□

Proposition 4.3. *Two finite ordered sets P and Q are order-isomorphic if and only if they can be drawn with identical Hasse diagrams.*

Proof:
The proof of this Lemma can be found at page 14 of [Davey and Priestley, 2002].□

The concept of ordered set is straightforwardly extended to the concept of ordered family

of sets; the following definition can be seen as the ordered version of Definition 4.8.

Definition 4.19.
Let A be a set, $\langle I, < \rangle$ a non-empty poset and C a collection of subsets of A. Then a bijective function $\mathcal{A} : \langle I, < \rangle \longrightarrow C$ is a ***partially ordered family*** *of subsets of A indicated with the following notation: $\{A_i\}_{i \in \langle I, < \rangle}$.*

As well as for non-ordered families, the notation $\{A_i\}_{i \in \langle I, < \rangle}$ is quite cumbersome and in the following we will adopt the shorthand notation to indicate a partially-ordered family: $\mathcal{A}_{\langle I, < \rangle}$.

From Definition 4.19 we understand that I and C are order-isomorphic and thus that $\mathcal{A}$ is an **order-isomorphism** such that $j < k$ in I if and only if $\mathcal{A}(j) < \mathcal{A}(k)$ in C. This means that $\forall j, k \in \langle I, < \rangle \mid j < k \Rightarrow \exists! \{A_j, A_k\} \in \mathcal{A}_{\langle I, < \rangle} \mid A_j < A_k$. It is worthwhile to notice that $A_j < A_k$ is not an inclusion order between the sets.

4.1.6 Metrics and Sets

There are many contexts in mathematics where one would like to say that two mathematical objects are close, and understand precisely what that means. For instance, if the two objects are the points (x_1, x_2) and (y_1, y_2) in a plane, then the task is quite straightforward: the Euclidean distance between them is: $\sqrt{(y_1 - x_1)^2 + (y_2 - x_2)^2}$, by the Pythagorean theorem, and it makes sense to say that the points are close if this distance is small. An important question is which are the properties that a definition of distance must have. A *metric space* is an abstract notion that results from thinking about this question. To this purpose, let X be a set of "points". Suppose that, given any two of these points x and y say, we have a way of assigning a real number $d(x, y)$ that we wish to regard as the distance between them, the following definition points out the properties that d has to respect to be a proper metric.

Definition 4.20.
Let X be a set, and $d \in \mathbb{R}$ a distance defined on a pair of elements (x, y) from X. Then, we define the distance d to be a ***proper metric*** *if, for any three elements $x, y, z \in X$, the following properties are respected:*

Identity. $d(x, x) = 0$.

Non-Negativity. $d(x, y) \geq 0$.

Symmetry. $d(x, y) = d(y, x)$.

Triangle Inequality. $d(x, y) + d(y, z) \geq d(x, z)$.

The first property is the identity notion: the distance between a point and itself must be equal to zero. The second says that the distance between two points is always positive, except

when the two points are the same, when it is zero (identity). The third property says that the distance is a symmetric notion: the distance between x and y is the same as the distance between y and x. The last is called triangle inequality: if you imagine x, y and z composing a triangle, it says that the length of any side never exceeds the sum of the lengths of the other two sides.

Then, a distance d defined on a pair of points (x, y) from a set X is called a proper metric if it respects these four properties. In that case, the pair (X, d) is a **metric space**.

The distance functions of metric spaces represent a way of quantifying the closeness of objects in a given domain – they establish how similar two objects are by defining a so-called *similarity measure* [Zezula et al., 2006]. A popular distance measure between sets is the *Jaccard's coefficient* [Jaccard, 1901] which defines the similarity between two sets.

Definition 4.21.
*Let A and B be two sets, then the **Jaccard's coefficient** is defined to be:*

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{4.11}$$

*and the **Jaccard's distance** to be:*

$$d_J(A, B) = 1 - J(A, B). \tag{4.12}$$

The distance measure is simply based on the ratio between the cardinalities of intersection and union of the compared sets. The Jaccard's distance (d_J) is a **proper metric**, satisfying: identity ($d_J(A, A) = 0$), non-negativity ($d_J(A, B) \geq 0$), symmetry ($d_J(A, B) = d_J(B, A)$) and triangle inequality ($d_J(A, B) + d_J(B, C) \geq d_J(A, C)$). The first three properties are trivially true [Clarkson, 2006], instead triangle inequality has been proved in [Lipkus, 1999; Spath, 1980].

Example 4.5. *Let us consider three sets $A_3 = \{c\}$, $A_4 = \{e, f, g\}$, and $A_5 = \{f, g\}$ – please see Figure 4.1b for a graphical representation of these sets. It is trivial [Spath, 1980] that:*

$$0 \leq J(A, B) = \frac{|A \cap B|}{|A \cup B|} \leq 1 \tag{4.13}$$

For instance,

$$J(A_3, A_4) = \frac{|\emptyset|}{|\{c, e, f, g\}|} = \frac{0}{4} = 0$$

and

$$J(A_4, A_5) = \frac{|\{f, g\}|}{|\{e, f, g\}|} = \frac{2}{3} = 0.66$$

From these examples we can say that the sets A_3 and A_4 are completely dissimilar – i.e.they are disjoint sets and their Jaccard's distance is 1 – and that the sets A_4 and A_5 are quite similar because their Jaccard's distance is $0.34 = 1 - 0.66$.

4.2 Graph Theory

4.2.1 A Historical Glance

Graph theory may be said to have its beginning in 1736 when Euler considered the (general case of the) Königsberg bridge problem. The East Prussian city of Königsberg (now Kaliningrad) occupies both banks of the River Pregel and an island, Kneiphof, which lies in the river at a point where it branches into two parts. There were seven bridges that spanned the various sections of the river, and the problem posed was this: could a person devise a path through Königsberg so that one could cross each of the seven bridges only once and return home? Long thought to be impossible, the first mathematical demonstration of this was presented by Euler to the members of the Petersburg Academy on August 26, 1735, and written up the following year under the title "*Solutio Problematis ad Geometriam Situs Pertinentis*" (The solution to a problem relating to the geometry of position) [Euler, 1736] cited by [Alexanderson, 2006].

"*The origins of graph theory are humble, even frivolous. Whereas many branches of mathematics were motivated by fundamental problems of calculation, motion, and measurement, the problems which led to the development of graph theory were often little more than puzzles, designed to test the ingenuity rather than to stimulate the imagination. But despite the apparent triviality of such puzzles, they captured the interest of mathematicians, with the result that graph theory has become a subject rich in theoretical results of a surprising variety and depth*" [Biggs et al., 1986].

It took 200 years before the first book on graph theory was written. This was "*Theorie der endlichen und unendlichen Graphen*" (Teubner, Leipzig, 1936) by König in 1936. Since then graph theory has developed into an extensive and popular branch of mathematics, which has been applied to many problems in mathematics, computer science, and other scientific and not-so-scientific areas [Biggs et al., 1986].

4.2.2 Graphs

A graph is a pair $G = (V, E)$ of sets such that $E \subseteq V \times V$; thus, the elements of E are 2-element subsets of V. The elements of $V = \{v_1, v_2, \ldots, v_n\}$ are called *vertices* of the graph G, the elements of $E = \{e_1, e_2, \ldots, e_m\}$ are called *edges* of the graph G. The vertex set of a graph G is referred to as $V(G)$, its edge set as $E(G)$.

An edge can be indicated as an unordered pair $e = \{v_i, v_j\}$ that means that e joins the vertices v_i and v_j; we indicate this vertex with the shorthand notation $e_{i,j}$. The two vertices *incident* with an edge are its *ends*, and en edge *joins* its ends. The set of all the edges at a vertex v is denoted by $E(v)$. The *degree* of a vertex v is the number $|E(v)|$ of edges at v.

Two vertices v_i, v_j of G are *adjacent*, if e_{v_i,v_j} is an edge of G. If all the vertices of G are pairwise adjacent, then G is *complete*.

As well as we have done in set theory, it is worthwhile to focus on the concept of isomorphism between graphs.

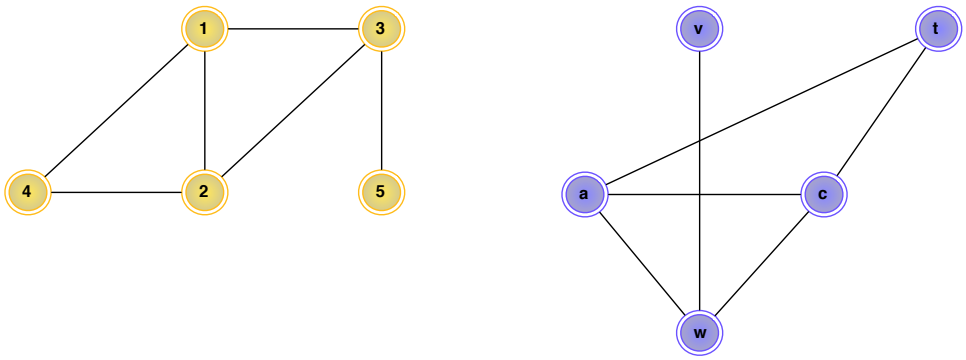


Figure 4.5: Two isomorphic graphs: $G(V, E)$ and $G'(V', E')$.

Definition 4.22.
*Let $G = (V, E)$ and $G' = (V', E')$ be two graphs. We define G and G' to be **isomorphic** if there exists a bijective function $\varphi : V \to V' \mid e_{i,j} \in E \Leftrightarrow e_{\varphi(i),\varphi(j)} \in E'$.*

In other words, we can relabel the vertices of G to be the vertices of G', maintaining the corresponding edges in G and G'. In the next example we can see two isomorphic graphs.

Example 4.6. *Let $G = (V, E)$ be an undirected graph where $V = \{1, 2, 3, 4, 5\}$ and $E = \{e_{1,2}, e_{1,3}, e_{1,4}, e_{2,3}, e_{2,4}, e_{3,5}\}$. Let $G' = (V', E')$ be an undirected graph where $V' = \{a, v, c, w, t\}$ and $E = \{e_{a,c}, e_{a,w}, e_{a,t}, e_{c,w}, e_{c,t}, e_{w,v}\}$. In Figure 4.5 we can see a graphical representation of G and G'. The function φ from V to V' given by $\varphi(1) = a, \varphi(2) = c, \varphi(3) = w, \varphi(4) = t, \varphi(5) = v$ is the required bijective function.*

4.2.3 Undirected and Directed Graphs

We consider two kinds of graphs: directed and undirected. In an **undirected graph** $G = (V, E)$, the edge set E consists of unordered pairs of vertices. We say that two vertices $v_i, v_j \in V$ are *connected* if exists an edge $e_{i,j} \in E$ that connect v_i and v_j; in an undirected graph the edge has no direction, v_i and v_j are not ordered and thus, $e_{i,j} = e_{j,i}$.

Given an undirected graph $G = (V, E)$, its **directed version** is the graph $G' = (V, E')$, where $e_{i,j} \in E'$ if and only if $e_{i,j} \in E$. That is, each undirected edge $e_{i,j}$ is replaced by the two directed edges $e_{i,j}$ and $e_{j,i}$. If it is not differently specified when we talk of a graph, we refer to a directed graph where an edge $e_{i,j}$ is denoted by its initial vertex i and its final vertex j.

A *path* is any sequence of edges where the final vertex of one is the initial vertex of the next one. In [Cormen et al., 2001] a path of length k from a vertex i to a vertex j in a graph $G = (V, E)$ is defined as a sequence $\langle v_0, v_1, \ldots, v_k \rangle$ of vertices such that $i = v_0$, $j = v_n$, and $e_{i-1,i} \in E$ for $i = 1, 2, \ldots, k$.

Definition 4.23.
*A **path** is a non-empty graph $P = (V, E)$ with $V = \{v_0, v_1, v_2, \ldots, v_k\}$ and $E = \{e_{0,1}, e_{1,2}, e_{2,3}, \ldots, e_{k-1,k}\}$, where all the v_i are distinct.*

The vertices v_0 and v_k are *linked* by P and are called its *ends*. The vertices $v_1, \ldots, v_{k-1}$ are the inner vertices of P. The number of edges of a path is its *length*, and the path of length k is denoted by P^k.

We often refer to a path by the natural sequence of its vertices, writing, say, $P = v_0 v_1 v_2 \ldots v_k$ and calling P a path from v_0 to v_k (as well as *between* v_0 and v_k). Sometimes it is useful to indicate such a path as: $v_0 P v_k$ [Diestel, 2006].

An *elementary path* is a path that does not use the same vertex more than once and a *simple path* is a path which does not use the same edge more than once.

Example 4.7. *Let us consider the graph $G(V, E)$ of Figure 4.5. We can point out a path $P = (V_P, E_P)$ where $V_P = \{1, 2, 3, 4, 5\}$ and $E_P = \{e_{1,4}, e_{4,2}, e_{2,3}, e_{3,5}\}$. As we can see the path P is a simple and elementary path with length 4 (P^4) and $|V_P| = 5$.*

If in a graph there is a path $v_i P v_j$ we say that v_j is *reachable* from v_i via P. An important concept is that of a *circuit* or *cycle*, which is a path where the initial and the final vertex are the same; a *loop* is a circuit comprising only one node. Formally, if we consider a path $P = (V, E)$ where $V = \{v_0, v_1, \ldots, v_{k-1}\}$, $E = \{e_{0,1}, \ldots, e_{k-2,k-1}\}$ with $k \geq 3$, then the graph $C = (V, E \cup \{e_{k,0}\})$ is called a *circuit*. A graph without circuits or loops is called an *acyclic graph*.

A graph is said to be *connected* if there exists a path from every couple of vertices in V and *complete* if there exists a path of length 1 for every couple of vertices in the graph.

Lastly, from the concept of path we can point out a very important property often called the *graphical distance* between two vertices.

Definition 4.24.
*Let $G = (V, E)$ be a graph and $v_i, v_j \in V$ be two vertices. The **graphical distance** $d_G(v_i, v_j)$ in G is the length of the shortest path from v_i to v_j in G.*

Example 4.8. *Let us consider the graph $G = (V, E)$ of Figure 4.5 and the path $P = (V_P, E_P)$ from 1 to 5. In the Example 4.7, we considered a path of length 4, but the shortest path $P = (V_P, E_P)$ from 1 to 5 is $V_P = \{1, 3, 5\}$ and $E_P = \{e_{1,3}, e_{3,5}\}$ where P^2. Thus, the graphical distance $d_G(1, 5) = 2$.*

4.2.4 Trees

One of the most important concept in graph theory, and one which appears often in areas superficially unrelated to graphs, is that of a tree. An acyclic graph is called a *forest* and a connected forest is called a *tree*; thus, a forest is a graph whose components are trees.

The definition of tree can be stated in different ways as it is described in [Christophides, 1975] where a tree is a *connected graph without a circuit* or a *graph in which every pair of vertices is connected with one and only one elementary path*.

Definition 4.25.
Let $T = (V, E)$ be a connected graph where $|V| = n$ and $|E| = (n - 1)$, then T is a ***tree****.*

When we consider a tree $T = (V, E)$ we call V the set of *nodes* instead of the set of vertices as we do with "generic" graph. The following proposition points out a set of equivalent assertions about trees.

Proposition 4.4. *Let $T = (V, E)$ be a graph, then the following assertions are equivalent:*

- *T is a tree.*
- *Any two nodes of T are linked by a unique path in T;*
- *T is minimally connected, i.e. T is connected but for every $e_{i,j} \in E, T = (V, E \setminus \{e_{i,j}\})$ is disconnected.*
- *T is maximally acyclic, i.e. T contains no cycle but for every $v_i, v_j \in V$ such that $\nexists e_{i,j} \in E$ then $T = (V, E \cup \{e_{i,j}\})$ is a cyclic graph and thus it is not a tree.*

Proof:
The proof of this well-known proposition can be found at page 14 of [Diestel, 2006].□

Rooted Trees

Often it is convenient to consider one node of a tree as special; such a node is then called the *root* of a tree. A tree $T = (V, E)$ with a fixed root $v_r \in V$ is a **rooted tree**. Let us consider a path $P = (V_P, E_P)$ from the root $v_r \in V$ to a node $v_k \in V$ such that $V_P \subset V$ and $E_P \subset E$, then for every $v_i, v_j \in V_P$ writing $v_i \prec v_j$ defines a partial order on V: the *tree-order* associated with T and v_r. We shall think of this ordering as expressing "*depth*": if $v_i \prec v_j$ we say that v_i lies below v_j in T. The nodes at distance $k \in \mathbb{N}$ from v_r have *depth k* and form the k^{th} *level* of T.

If it is not further specified when we talk of a tree we refer to a rooted tree. In a rooted tree there is the explicit concept of hierarchical relation between the nodes of the tree; indeed, if v_j is said to lie below the node v_i, it means that v_i is hierarchically superior to v_i. In an rooted tree the edges are *ordered pairs* such as $e_{i,j} = (v_i, v_j) \in E$ means that v_i is linked to v_j by this edge

and that v_i is *higher* than v_j (or that v_j is deeper than v_i). If there exists $e_{i,j} \in E$ then it meaning that we can go downward from v_i to v_j and upward from v_j to v_i. In this case we define v_i to be the **parent node** of v_j and vice versa, v_j to be a **child node** of v_i. In a tree a node has at most one parent but can have no, one or more children. From the notion of path we can point out that a node v_k is said to be an **ancestor** of a node v_j if there exists a path from v_k to v_j; on the other hand, v_j is said to be a **descendant** of v_k. Please note that every node is both an ancestor and a descendant of itself.

In this context it is convenient to talk about *inbound edges* and *outbound edges* of a node.

Definition 4.26.
Let $T = (V, E)$ be a rooted tree and $v_i \in V$ be a node of the tree, then we define its:

Inbound set *to be* $E^-(v_i) = \{v_j \in V \mid e_{j,i} \in E\}$.

Outbound set *to be* $E^+(v_i) = \{v_j \in V \mid e_{i,j} \in E\}$.

Inbound degree *to be* $|E^-(v_i)|$.

Outbound degree *to be* $|E^+(v_i)|$.

In the definition above there is no risk of ambiguity and thus, we omitted the subscripts in the notation regarding the inbound/outbound sets; otherwise, for example, we shall indicate the inbound set of the node $v_i \in V$ in the tree $T = (V, E)$ as $E^-_V(v_i)$. Let us consider the rooted tree $T = (V, E)$; the inbound set $E^-(v_i)$ is the set of all the nodes that are *higher* than $v_i \in V$ and linked to it by an edge in E. The outbound set $E^+(v_i)$ is the set of all the nodes that are *deeper* than $v_i \in V$ and linked to it by an edge in E. The correspondent degrees are the cardinalities of these sets.

The inbound and outbound degrees are well-suited to define the properties of the nodes of the rooted trees. Indeed, $v_r \in V$ is defined to be the **root** of $T = (V, E)$ then $|E^-(v_r)| = 0$ and $\forall v_i \in V \setminus \{v_r\}$, $|E^-(v_i)| = 1$. The set of all **external nodes** or **leaves** is $V_{ext} = \{v_i \in V \mid |E^+(v_i)| = 0\}$ and the set of all the **internal nodes** is $V_{int} = \{v_i \in V \mid |E^+(v_i)| > 0\}$.

Furthermore, we define two additional sets of nodes that are very useful in the rest of the work. We define with $\Gamma^+(v_i)$ the set of **all the descendants** of v_i in V (including v_i itself); vice versa $\Gamma^-(v_i)$ is the set of **all the ancestors** of v_i in V (including v_i itself). Let us consider a rooted tree $T = (V, E)$, then for all $v_j \in \Gamma^+(v_i)$ it exists a path $P = (V', E')$ from v_i to v_j in T, where $|E'| > 0$. Vice versa, for all $v_j \in \Gamma^-(v_i)$ it exists a path $P = (V', E')$ from v_j to v_i in T, where $|E'| > 0$.

Let us consider a couple of recurrent cases: let $v_r \in V$ be the root of a tree $T = (V, E)$ then $\Gamma^-(v_r) = \{v_r\}$ and $\Gamma^+(v_r) = V$. Furthermore, let v_i an external node of $T = (V, E)$, then $\Gamma^+(v_i) = \{v_i\}$.

The following example shows how these concept are applied to a sample tree.

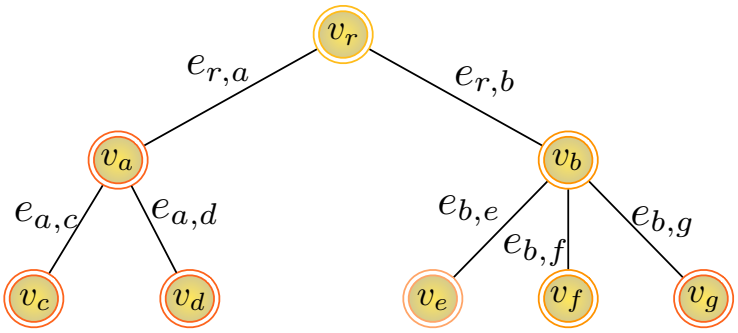


Figure 4.6: A sample tree $T(V, E)$.

Example 4.9. *Let $T(V, E)$ be a tree where $V = \{v_r, v_a, v_b, v_c, v_d, v_e, v_f, v_g\}$ and $E = \{e_{r,a}, e_{r,b}, e_{a,c}, e_{a,d}, e_{b,e}, e_{b,f}, e_{b,g}\}$. In Figure 4.6 we can see a graphical representation of this tree.*

For instance, the path $P = (V_P, E_P)$ from v_r to v_e has $V_P = \{v_r, v_b, v_e\}$ and $E_P = \{e_{r,b}, e_{b,e}\}$, thus it has length is 2. The set $E^+(v_a) = \{v_c, v_d\}$, $E^-(v_a) = \{v_r\}$, $E^-(v_r) = \{\emptyset\}$ and $E^+(v_e) = \{\emptyset\}$. The inbound degree of the node v_r is $|E^+(v_r)| = 0$ instead $|E^+(v_a)| = 1$. The outbound degree of v_a is $|E^+(v_a)| = 2$ instead $|E^+(v_e)| = 0$.

$V_{ext} = \{v_c, v_d, v_e, v_f, v_g\}$ and $V_{int} = \{v_r, v_a, v_b\}$. Finally, the set of all the descendants of v_a is $\Gamma^+(v_a) = \{v_a, v_c, v_d\}$ and the set of all its ancestors is $\Gamma^-(v_a) = \{v_a, v_r\}$, please note that the node v_a is included in both of these sets.

From this example we can see that we can define a **subtree** rooted at v_a induced by the descendants of v_a in V. The subtree rooted in v_a in Figure 4.6 contains the nodes v_a, v_c and v_d.

Furthermore, by means of this newly described notation, we can formally define the important concept of **lowest common ancestor** that we introduced in Section 2.2.3. The lowest common ancestor of nodes v_j and v_k in a tree is the ancestor of v_j and v_k that is located farthest from the root [Bender et al., 2005].

Definition 4.27.
*Let $T(V, E)$ be a tree and $v_j, v_k \in V$ be two vertices. Then we define v_t to be the **lowest common ancestor** of v_j and v_k ($\mathsf{lca}(v_j, v_k) = v_t$) if:*

$$v_t \in \Gamma^-(v_j) \cap \Gamma^-(v_k), \text{ and} \tag{4.14}$$

$$\nexists v_w \in V, w \neq t \mid (v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k)) \wedge (v_w \in \Gamma^+(v_t)) \tag{4.15}$$

The first condition imposes that $v_t = \mathsf{lca}(v_j, v_k)$ must be a common ancestor of v_j and v_k; the second condition says that cannot exist a vertex that is not v_t which is nearer than v_t to both v_j and v_k.

Ordered Trees

An **ordered tree** is a rooted tree in which the children of each node are ordered; that is, if a node has k children, then there is a first child, a second child and so on and so forth. When we consider ordered trees we take into account also the horizontal dimension of the hierarchy and thus the relationships between the sibling nodes. A special case of ordered tree is the **binary tree** that is a tree that contains no nodes or that is composed by three disjoint sets of nodes: a root node, a binary tree called its **left subtree**, and a binary tree called its **right subtree** [Cormen et al., 2001].

We use the following notation to indicate an ordered tree: $T = (\langle V, < \rangle, E)$ where V is the poset of vertices. $\langle V, < \rangle$ defines a partial order between the vertices of an ordered tree and from the common definition of ordered tree, only the order between siblings is important for practical purposes.

4.2.5 The Metric Properties of Trees

The distance between two nodes is a fundamental concept for trees; we can define the distance between two nodes belonging to the same tree or the distance between two trees. In the former case we talk about how many edges we have to cross to reach a node from another one. In the latter case we are defining a similarity measure where the more two trees are distant, the more they differ one from the other [Zezula et al., 2006].

Let $T = (V, E)$ be a tree, then the distance between two vertices v_i and v_j in V can be seen as the number of edges that we need to cross to go from v_i to v_j; this is the *graphical distance* (d_G) between two nodes. If we consider the same tree T where each edge is associated with a length 1, then the pair (T, d_G) is a **metric space** [Buneman, 1974; Gowers et al., 2008]. The notion of graphical distance is important in graph theory and in tree algorithms; furthermore, many metric properties of tree are defined using the concept of graphical distance.

A meaningful example is the *four-point condition* where by checking the possible configurations of paths which can connect four nodes $x, y, z, t \in V$ in a tree $T = (V, E)$, it has been proven [Buneman, 1974] that the graphical distance must satisfy:

$$d_G(x, y) + d_G(z, t) \leq \max\{d_G(x, z) + d_G(y, t),\ d_G(x, t) + d_G(y, z)\} \tag{4.16}$$

The four-point condition is stronger than the triangle inequality and is equivalent to say that of the three sums $d_G(x, y) + d_G(z, t)$, $d_G(x, z) + d_G(y, t)$, and $d_G(x, t) + d_G(y, z)$ two are equal and not less than the third. The four-points condition is also sufficient condition to a graph to be a tree as pointed out by the following lemma:

Lemma 4.5. *A graph is a tree if and only if it is connected, contains no triangles, and has graphical distance satisfying the four-point condition.*

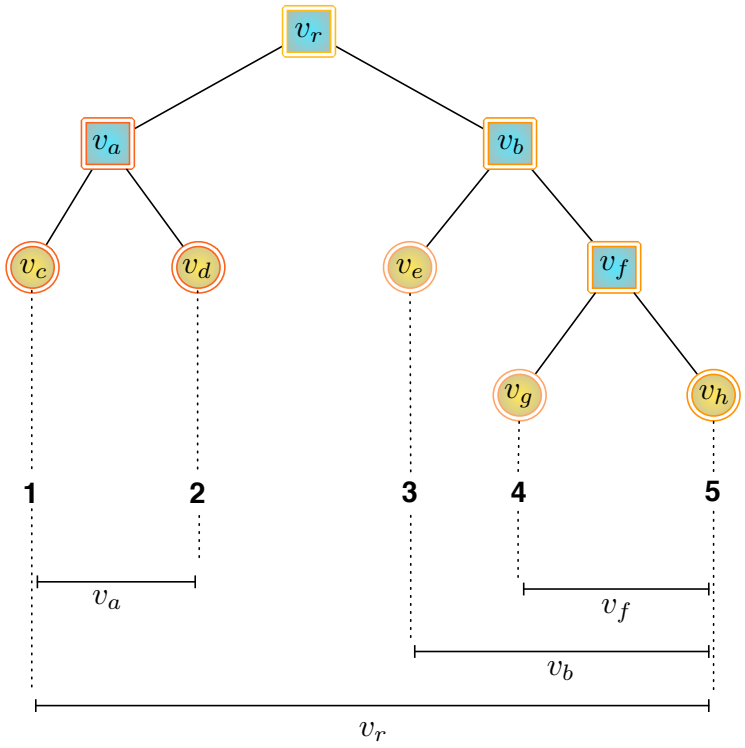


Figure 4.7: The intervals associated with the internal nodes of a rooted ordered tree.

Proof:
The proof of this lemma can be found in [Buneman, 1974].

The notion of distance can be applied also between two trees to establish how close or distant they are. Often these distance measures are called *similarity* or *dissimilarity measures*.

A widely-used distance measure between trees is the *tree edit distance* [Apostolico and Galil, 1997; Sankoff and Kruskal, 1983]. The tree edit distance defines a distance between two tree structures as the minimum cost needed to convert a tree to another tree using a predefined set of edit operations: insertion, deletion and relabeling of a node; indeed, the problem of computing the distance between two trees is associated to labeled trees. An extensive survey about the algorithms to compute the tree edit distance is [Bille, 2005]. Since XML documents are typically modeled as rooted labeled trees [Zezula et al., 2006], the tree edit distance can also be used to measure the structural dissimilarity of XML documents [Cobena et al., 2002; Guha et al., 2002].

Another meaningful structural distance between trees is the Parseval metric [Abney et al., 1991] extended by Gallé in [Gallé, 2010]. This distance is based on the concept of "brackets of tree" which is a set of intervals that permits us to reconstruct the tree from a set of values. Let us consider the rooted ordered tree in Figure 4.7 to understand how these intervals are defined. Every external node – i.e. v_c, v_d, v_e, v_g, and v_h – is associated with an integer and every internal

node – i.e. v_r, v_a, v_b and v_f – is associated with an interval defined by the smaller integer and the bigger integer of the external nodes it comprises; so the interval set associated with a tree is composed by the intervals associated to the internal nodes. For instance, the internal node v_r is the root and thus, it comprises all the external nodes in the tree and its interval is defined by the smaller integer – i.e. 1 – which is associated to v_c and the bigger – i.e. 5 – which is associated to v_h. Following this procedure the interval set is $X = \{[1, 5], [1, 2], [3, 5], [4, 5]\}$. As we can see $|X| = |V_{int}| = 4$. The structure of a tree can be fully recovered by its interval set (alternatively called bracket set) [Abney et al., 1991; Wojcik et al., 1993].

The structural distance between two trees is computed comparing the interval set associated to each tree. So, let us assume that a rooted ordered tree $T_X = (V_X, E_X)$ is associated to the interval set X and that the rooted ordered tree $T_Y = (V_Y, E_Y)$ is associated to the interval set Y. In [Gallé, 2010] the following *assignment function* is defined:

$$f_a : X \to Y \cup \{\emptyset\}.$$

This function is required to be injective, this means that every interval in Y has at most one corresponding interval in X [2]; please note that $\text{ran}(f_a) \subseteq Y$. The role of the empty set in the image is to permit to assign brackets from X which otherwise would not be assigned. If for any $x \in X, f_a(x) = \emptyset$ we refer to this as a *null-assignment*. The **structural distance** between two trees is defined as follows [Gallé, 2010]:

$$d_S(X, Y) = \sum_{x \in X} d(x, f_a(x)) + |Y \setminus \text{ran}(f)| \tag{4.17}$$

This gives a penalty of a maximal distance for every interval of Y to which no interval of X was assigned. This is the symmetric part of assigning $\emptyset$ to an interval of X. This distance is proved to be a proper metric in [Gallé, 2010].

[2] Please note that an injective function preserves distinctness: it never maps distinct elements of its domain to the same element of its co domain.

Chapter 5

Digital Libraries

In this chapter we present the main characteristics of Digital Libraries with a particular attention to the archives that represent the foremost use case analyzed in this work. Furthermore, we describe the standard digital library technologies we exploit in this work.

5.1 Overview on Digital Libraries

The term "Digital Libraries" corresponds to a very complex notion with several diverse aspects and it cannot be captured by a simple definition [Candela et al., 2007b]. Indeed, the term "digital libraries" is used to refer to systems that are very heterogeneous in scope and provide very different functionalities [Candela et al., 2007a]. These systems span from digital object and metadata repositories, reference-linking systems, archives, and content administration systems, to complex systems that integrate advanced services. Furthermore, digital libraries represent the meeting point of many disciplines and research fields – i.e. data management, information retrieval, library and information sciences, document and information systems, the Web, information visualization, artificial intelligence, human-computer interaction, and others [Ioannidis, 2005].

Thus, providing a unique definition of "digital libraries" is quite problematic and every attempt with this aim has led to an application-oriented definition; indeed many tentative definitions are "*influenced by the primary discipline of their proposer(s)*" [Candela et al., 2007b]. Indeed, in [Ioannidis, 2005] digital libraries are envisioned as "citizen-oriented" systems: "*digital libraries should enable any citizen to access all human knowledge, any time and anywhere, in a friendly, multi modal, efficient and effective way, by overcoming barriers of distance, language, and culture and by using multiple Internet-connected devices.*" Likewise, in [Soergel, 2002] a research framework on digital libraries has been proposed starting from "*three very different perspectives that different people in the community have on digital libraries, i.e., as tools to serve research, scholarship, and education, as a means for accessing information, and as providing services primarily to individual users*". On the other hand, in [Belkin, 1999] we can find a narrower view of digital libraries intended as an: "[...] *institution in charge of providing at least the functionality of a traditional library in the context of distributed and net-*

worked collections of information objects". In [Borgman, 1999] there is a distinction between at least two competing visions of the expression "digital library": "*Researchers view digital libraries as content collected on behalf of user communities, while practicing librarians view digital libraries as institutions or services.*" The vision proposed by the "DELOS Network of Excellence on Digital Libraries"[1] catches in a proper way the difficulties in defining the boundaries of digital libraries stating that: "*digital library* [*is*] *a tool at the centre of intellectual activity having no logical, conceptual, physical, temporal, or personal borders or barriers on information.*"

Despite this multifaceted view of digital libraries there are several accepted conceptions that are shared by the researchers of the area:

- *User-centric systems*: Digital libraries are a new type of information infrastructures that should be user-centered, able to support content management tasks together with tasks devoted to communication and cooperation. That is they should be information infrastructures that become common vehicles with which every user can access, discuss, evaluate, and enhance information of all forms. Although they are still places where information resources can be stored and made available to end users, the current design and development efforts are moving in the direction of transforming them into infrastructures able to support the user in different information centric activities.

- *Dynamic interactions*: The vision of digital libraries has evolved from static storage and retrieval of information to dynamic forms of facilitation of communication, collaboration and other forms of interaction among scientists, researchers or the general public.

- *Large capabilities*: Digital libraries evolve from handling mostly centrally located textual documents to synthesizing distributed multimedia document collections, sensor data, mobile information, and pervasive computing services.

Digital libraries have contributed to supporting the creation of innovative applications and services to access, share and search our cultural heritage. In this context, another key feature we have to consider to understand the world of digital libraries is that they have to take into account several distributed and heterogeneous information sources with different community background and different information objects ranging from full content of digital objects to the metadata describing them. These objects can be exchanged between distributed systems or they can be aggregated and accessed by users with distinct information needs and living in different countries. Indeed, one of the most important contributions of digital libraries is to make available collections of digital resources from different cultural institutions such as *libraries*, *archives* and *museums*, to make them accessible in different languages and to provide

[1] DELOS: Network of Excellence on Digital Libraries was co-funded by the European Union. DELOS run for ten years (3 years as a working group, 3 years as a Thematic Network and 4 years as a Network of Excellence) and it has given a substantial contribution to the establishment in Europe of a research community on digital libraries. Now there is the "*DELOS Association*" that is a not-for-profit organization whose main aim is to continue as much as possible the DELOS activities. `http://www.delos.info/`

advanced services over them. We have to consider that the above mentioned institutions are different from several point-of-views: their internal organization has different peculiarities, the resources they collect and manage have different structure and nature, these resources are described with different means and for different purposes, their users have different information needs and require different methods to access the resources. Thus, digital libraries are heterogeneous systems with peculiarities and functionalities that range from data representation to data exchange and data management. Furthermore, digital libraries are meaningful parts of a global information network which includes scientific repositories, curated databases and commercial providers. All these aspects need to be taken into account and balanced to support final users with effective and interoperable information systems. A fundamental role of digital libraries therefore is to provide data models, protocols, applications and services to handle all these cultural resources preserving their characteristics and addressing the issues related to their differences.

Digital libraries can be defined and shaped also by analyzing the issues they have to address; in particular, two important aspects that digital libraries have to study are: *heterogeneity* – which we have just described – and *interoperability*. These two concepts are closely-related and their interrelation can be pointed out particularly by analyzing interoperability as a complex and multiform concept, which can be defined - as by the "ISO/IEC 2382-01, Information Technology Vocabulary, Fundamental Terms" - as follows: "*The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units*". When the concept of interoperability is considered in the context of digital libraries it takes on different dimensions as shown by the *European Commission Working Group on digital library Interoperability* [Gradmann, 2007] which identified six dimensions that can be distinguished and taken into account:

Interoperating entities. These can be assumed to be the traditional cultural heritage institutions, such as libraries, museums, archives, and other institutions in charge of preservation of artifacts or that offer digital services.

Information objects. The entities that actually need to be processed in interoperability scenarios. Choices range from the full content of digital information objects to the metadata describing them.

Functional perspective. This may simply be the exchange and/or propagation of digital content. Other functional goals are aggregating digital objects into a common content layer.

Multilingualism. Linguistic interoperability can be thought of in two different ways: as multilingual user interfaces to digital library systems or as dynamic multilingual techniques for exploring the digital library systems object space.

User perspective. Interoperability concepts of a digital library system manager differ substantially from those of a content consuming end user.

Interoperability technology. Enabling different kinds of interoperability constitutes a major dimension and several technologies designed in the context of digital libraries.

These dimensions are particularly well-suited for understanding the compound world of digital libraries. The first point, "*Interoperating entities*" takes into account the nature of the institutions that are constituent parts of a digital library. Libraries, archives and museums have different organizations, traditions and internal policies; the resources produced and preserved by them reflect their very nature and thus they are quite different in structure and content. Furthermore, resources coming from different cultural environments have to be treated in different ways in terms of their access, management, preservation, and the modalities offered to the user for the fruition of the contents.

The second point "*information objects*" lets us understand the range of resources that digital libraries have to deal with. Full content digital information objects range from full text to multimedia objects that have to be related to the metadata describing them. Metadata – literally "data about data" – are fundamental resources in digital libraries that are used to described digital and traditional resources, and as an aid to managing, accessing, and retrieving them. Metadata are semi-structured data which describe the characteristics of a resource. They share many similar characteristics with the cataloging that takes place in libraries, museums and archives.

The third point "*functional perspective*" underlines the fact that digital libraries have to share their resources in a distributed environment and give access to them. On the other hand, the aggregation of resources in a common content layer is another key aspect of digital libraries. Digital libraries operate in an open and multilingual environment and the fourth point "*multilingualism*" highlights the importance of this aspect.

The fifth point "*User perspective*" is particularly relevant. Digital libraries have to take care of the all information life cycle that begins with a user request; thus, a prime goal of digital libraries is to design and develop languages and interfaces well-suited to the user interaction. Often *languages* developed in other fields – e.g. databases – are relatively narrow and cannot capture the full range of requests made to digital libraries [Ioannidis et al., 2005]. Thus, it is important to develop languages which are expressive enough for users to pose their most sophisticated needs succinctly. Digital libraries have to also take into account interaction paradigms providing personalized interactions which consider the "*diversity of backgrounds, needs, and preferences*" [Ioannidis et al., 2005] of the users.

The sixth point "*interoperability technologies*" is another key aspect that must be taken into account when we consider digital libraries. A common goal in the design and development of digital libraries is to build systems which rely as much as possible on existing building blocks, thus maximizing the exploitation of Web and Internet standards. When we deal with digital libraries we have to consider different technologies such as the metadata schema adopted to describe the resources, the encoding of the metadata schema, the data management system, the protocols to exchange metadata in distributed environments and, the means to relate metadata and full content digital object in a consistent way.

These are the main aspects that we take into account when we talk about digital libraries. As we have underlined before the world of digital libraries is very vast and complex, thus these aspects should be considered just as a meaningful point-of-view on this world. Several other facets of digital libraries can be taken into account depending on which research aspect we are aiming to investigate. For instance, digital libraries are also concerned with information and process quality aspects [Fuhr et al., 2007], information integration and derivation which is concerned with putting the information in the form most useful to the user [Ioannidis et al., 2005], and information enrichment which involves annotations [Agosti et al., 2005, 2007a; Agosti and Ferro, 2007; Ferro, 2009; Ferro and Silvello, 2010], provenance [Buneman and Tan, 2007; Castelli et al., 2010; Moreau et al., 2008], lineage [Simmhan et al., 2005] and citation [Altman and King, 2007; Buneman, 2006; Buneman and Silvello, 2010; Lawrence et al., 1999].

5.2 Archives

The context represented by the digital libraries is multifaceted and comprises many realities of interest. In this work we consider digital libraries but in particular, we focus on a specific and meaningful reality treated by them: the archives. We take archives into account because they are quite complex entities, organization and structure of which as well as the problems that have to be addressed at a digital level are well-suited for our research purposes.

Archives differ from libraries in the nature of the materials held. Libraries collect individual published books and serials, or bounded sets of individual items. The books and journals held by libraries are *not unique*, since multiple copies exist and any given copy will generally prove as satisfactory as any other copy. The material in archives and manuscript libraries are "*the unique records of corporate bodies and the papers of individuals and families*" [Pitti and Duff, 2001].

Moreover, an archive is not simply constituted by a series of objects that have been accumulated and filed with the passing of time. Instead, it represents the trace of the activities of a physical or juridical person in the course of their business which is preserved because of their continued value. Archives have to keep the context in which their records[2] have been created and the network of relationships between them in order to preserve their informative content and provide understandable and useful information over time.

Archival description is defined in [Pearce-Moses, 2005] as "*the process of analyzing, organizing, and recording details about the formal elements of a record or collection of records, to facilitate the work's identification, management, and understanding*"; archival descriptions have to reflect the peculiarities of the archive, retain all the informative power of a record, and keep trace of the provenance and original order in which resources have been collected and filed by archival institutions [Gilliland-Swetland, 2000]. This is emphasized by the central con-

[2] In [MacNeil et al., 2001] a record is defined as: "*Any document made or received and set aside in the course of a practical activity*".

cept of *fonds*[3], which should be viewed primarily as an "intellectual construct", the conceptual "whole" that reflects an organic process in which a records creator produces or accumulates series of records [Cook, 1993]. In this context, provenance becomes a fundamental principle of archives often referred to as "*respect des fonds*" which dictates that resources of different origins be kept separate to preserve their context [Duranti, 1998; Gilliland-Swetland, 2000].

[Duranti, 1998] highlights that maintaining provenance leads archivists to evaluate records on the basis of the importance of the creator's mandate and functions, and fosters the use of a hierarchical method. The hierarchical structure of the archive expresses the relationships and dependency links between the records of the archive by using what is called the archival bond defined as "*the interrelationships between a record and other records resulting from the same activity*" [Pearce-Moses, 2005]. Archival bonds, and thus relations, are constitutive parts of an archival record: if a record is taken out from its context and has lost its relations, its informative power would also be considerably affected. Therefore, archival descriptions need to be able to express and maintain such structure and relationships in order to preserve the context of a record. These aspects are highlighted if we consider the definition of "archival description" given by the glossary of the *International Standard for Archival Description (General)* (ISAD(G)) [International Council on Archives, 1999b] drawn up by the *International Council on Archives (ICA)*[4] for the development of archival information systems: "*an accurate representation of a unit of description and its component parts, if any, by capturing, analyzing, organizing and recording information that serves to identify, manage, locate and explain archival materials and the context and records systems which produced it.*"

The definition underlines that archives are complex entities; indeed, they are made up of a collection of entities and relations which link them to each other. It is precisely the nature of these links – i.e. archival bonds – which distinguishes archives from other objects in the realm of cultural heritage – e.g. books – which in general are perceived as individual, repeatable and unrelated entities [Vitali, 2010]. Archives are in fact made up of series which in turn can be organized in sub-series which are formed of archival units – e.g. files, registers, and so on – which have a homogeneous nature and can in turn be divided into subunits containing items such as letters, reports, contracts, testaments, photographs, drawings, etc.

Following this structure and according to ISAD(G), archival description proceeds from general to specific as a consequence of the provenance principle and has to show, for every unit of description, its relationships and links with other units and to the general fonds. Therefore, archival descriptions produced according to the ISAD(G) standard take the form of a tree which represents the relationships between more general and more specific archive units going from the root to the leaves of the tree. In Figure 5.1 we can see the ISAD(G) hierarchical model as it is represented in [International Council on Archives, 1999b]; any number of intermediate levels are possible between any shown in the model.

Entities are in a vertical relationships of subordination with the entity they belong to; the

[3]The term *fonds* is not a commonly used English word but it is derived from the French [Hayworth, 1993] and in the archival context it is used both for the singular and plural form of the noun.

[4]`http://www.ica.org/`

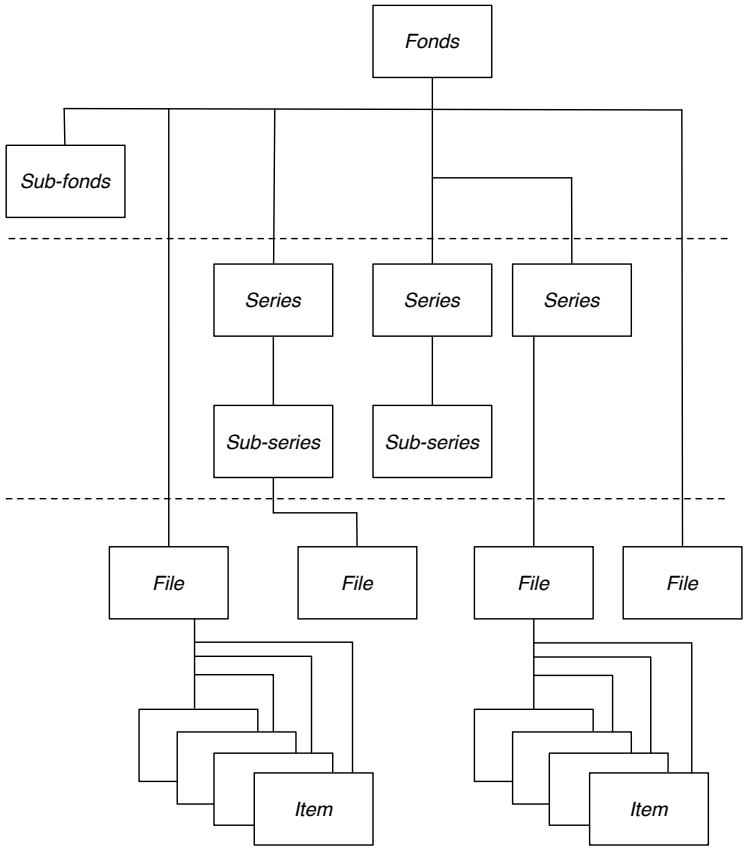


Figure 5.1: Hierarchical organization of the archives and of the archival descriptions according to ISAD(G) [International Council on Archives, 1999b].

hierarchical representation is further complicated by the fact that the entities which belong to the same father have a "horizontal-type" relationship – they need to be represented according to a significant sequence which reflects the position that they have in the logical and/or in the material order of the archive; thus, the hierarchical organization of archives can be defined as a "level hierarchy" according to the classification of hierarchies we presented in Section 2.1.1.

We have highlighted that the informational power of an archival record is given by the record itself plus its context determined by the set of relationships that it establishes with the other records in the archive. However, a similarly fundamental role in archival descriptions is played by other type of contexts, which in a certain sense are external to the archives themselves. Generally-speaking archives are the outcome of the practical activities of certain subjects such as corporate bodies, families or persons. These entities are called the creators of the archive and they are a constituent part of its context; the archival descriptions have to maintain the relationships also with these entities that in turn are described by means of other descriptive standards. These descriptions are called archival authority records associated with the creation and maintenance of archives. The ICA developed a descriptive standard for the redaction of these descriptions which is the *International Standard Archival Authority Records (Corporate Bodies, Persons, Families)* or ISAAR(CPF) [International Council on Archives,

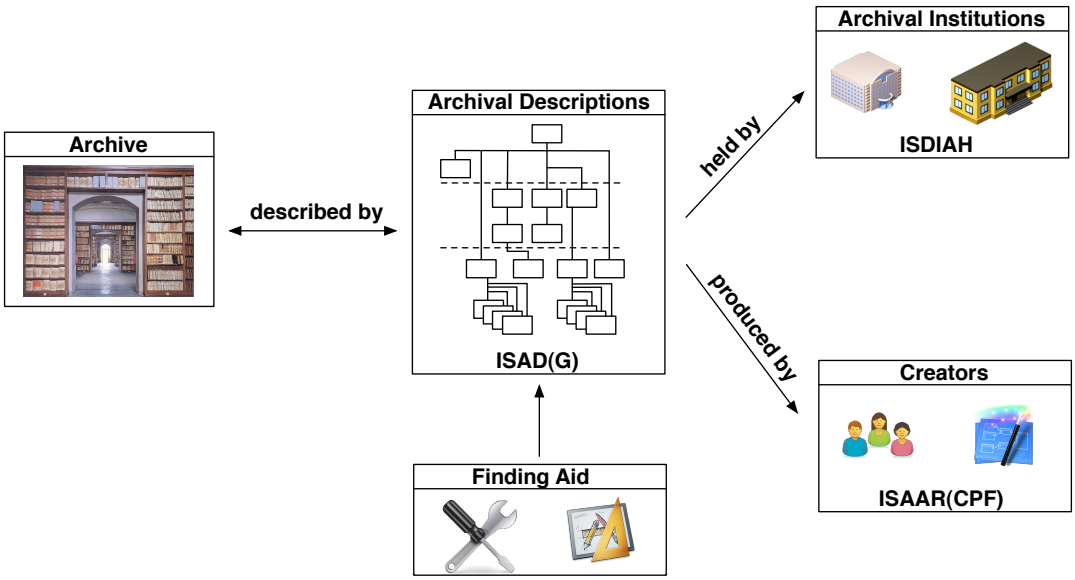


Figure 5.2: A graphical representation of the relationships between the archival descriptions.

1999a]. ISAAR(CPF) specifies that the archival authority records may be used to:

- describe a corporate body, person, or family as units within an archival descriptive system.
- Control the creation and use of *access points* in archival descriptions; the name of the creator of the unit of description is one of the most important of such access points.
- Document relationships between different record creators and between those entities and the records created by them and/or other resources about or by them.

Each archive therefore should be related with one or more creators who presided over its accumulation and their history, functions, activities, etc. should be described. Furthermore, when archives held by archival institutions are described in the same information system, these institutions also have to be described to help users locate the archives described [Vitali, 2010]; the ICA developed a standard for the description of archival institutions called *International Standard for Describing Institutions with Archival Holdings* (ISDIAH) [International Council on Archives – Committee of Best Practices and Standards (CBPS), 2008]. Other entities can be added to these such as the description of the finding aid[5] existing for a certain fonds, bibliographic references, other information resources, etc. with the outcome of creating relatively complex systems. In Figure 5.2 we can see a graphical representation of the components composing an archive and its descriptions.

[5]A finding aid is an essential access tool for conveying information about the arrangement and content of archives. Such finding aids often contain far more information about a collection than can be captured in a summary catalog record; they are generally created in the course of processing a collection and usually reflect the *hierarchical arrangement* of the materials [Ruth, 2001].

5.3 Digital Library Technologies

The role of the technology of choice is fundamental in the context of digital libraries. We have seen that technologies are one of the key aspects of digital library interoperability dimensions [Gradmann, 2007]. Digital libraries rely on standard technological building blocks that must be taken into account when dealing with them. The first technological building block is constituted by the *metadata schema* adopted to describe the resources managed by a digital library; we consider two main metadata schema: the Dublin Core (DC)[6] metadata initiative which is the standard in the library context and provides simple standards to facilitate the finding, sharing and management of information, and the Encoded Archival Description (EAD)[7] which is the standard metadata format in the archival context. While the syntax is not strictly part of the metadata schema, the data will be unusable, unless the encoding scheme understands the semantics of the metadata schema. The encoding allows the metadata to be processed by a computer program. Important schemes that we have to take into account include XML[8] and *Resource Description Framework (RDF)*[9].

The second technological building block is constituted by the *Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)*[10] which is the standard *de-facto* for metadata exchange in distributed environments.

The third technological building block we take into account is the *Open Archives Initiative - Object Reuse and Exchange (OAI-ORE)*[11] which provides a model for handling compound digital objects and their aggregations on the Web. Archives and museums should adopt these technological means to exploit the services offered by the digital libraries and to promote a deeper integration between the resources coming from different cultural environments.

Dublin Core Metadata Initiative

The *Dublin Core Metadata Initiative* (DCMI) is an international effort designed to foster consensus across disciplines for the discovery-oriented description of diverse resources in an electronic environment. The DCMI has, in part and to this end, defined the *Dublin Core Element Set* (DCES) which is intended to support cross-discipline resource discovery. The DCES is a vocabulary of fifteen properties for use in resource description: *Title, Creator, Subject, Descriptions, Publisher, Contributor, Date, Type, Format, Identifier, Source, Language, Relation, Coverage, and Rights*. These elements make up the "*simple Dublin Core*" which has been formally endorsed in the following standards: *ISO Standard 15836:2009 of February 2009*, *ANSI/NISO Standard Z39.85-2007 of May 2007*[12], and *IETF RFC 5013 of August 2007*[13].

[6] http://www.dublincore.org/
[7] http://www.loc.gov/ead/
[8] http://www.w3.org/XML/
[9] http://www.w3.org/RDF/
[10] http://www.openarchives.org/pmh/
[11] http://www.openarchives.org/ore/
[12] http://www.niso.org/standards/z39-85-2007/
[13] http://www.ietf.org/rfc/rfc5013.txt

The Dublin core intended to facilitate the description and discovery of electronic resources and since its first developments it pursues five main objectives [Weibel, 1997]: Simplicity, semantic interoperability, international consensus, flexibility and modularity.

As a core set of concepts, the DCES is not intended to satisfy every possible declarative need in every discipline; indeed, when it comes to metadata it is almost impossible to define a format which can be adopted in different application environments. Every application context has it own peculiarities and characteristics that need to be caught by a metadata; in such contexts we need to be able to specify new properties of a metadata format or a totally new schema. To this purpose the DCMI developed the *Dublin Core Application Profiles* (DCAP). A DCAP defines metadata records which meet specific application needs while providing semantic interoperability with other applications on the basis of globally defined vocabularies and models. A DCAP is a document (or set of documents) that specifies and describes the metadata used in a particular application. DCAP are very useful because they permit us to define new metadata format based on the principles that guided the development of the simple Dublin Core and thus enhancing the interoperability and flexibility aspects.

As a general statement we can say that the Dublin Core metadata format – comprising the possibility of designing and develop a DCAP – is tiny, easy-to-move, shareable and remarkably suitable for a distributed environment. These characteristics have made the Dublin Core one of the most diffuse and recognized metadata standards embraced by many realities.

Encoded Archival Description

The EAD is a standardized electronic representation of archival description which makes it possible to provide union access to detailed archival descriptions and resources in repositories distributed throughout the world. The development of the EAD started from a project of the University of California, Berkeley in 1993 which had the following important goal: "*investigate the desirability and feasibility of developing a non proprietary encoding standard for machine-readable finding aids such as inventories, registers, indexes, and other documents created by archives, libraries, museums, and manuscript repositories to support the use of their holdings*"[14].

The development of EAD continued over time and brought the first release of EAD Document Type Definition (DTD) by the Library of Congress in conjunction with the Society of American Archivists in 1998. Currently, EAD is in its second version released in 2002 and the entire suite of DTD and entity reference files are re engineered to meet the needs of XML and related technologies. Since the first release, EAD has been conceived in order to meet five main requirements, which are the ability to:

1. Present extensive and interrelated descriptive information found in archival finding aids.

2. Preserve the hierarchical relationships existing between levels of archival description.

[14] `http://www.loc.gov/ead/eaddev.html`

3. Represent descriptive information that is inherited by one hierarchical level from another.

4. Move within a hierarchical informational structure.

5. Support element-specific indexing and retrieval.

EAD rapidly became a standard for encoding archival descriptions; firstly, because it is based on an internationally recognized standard for markup language: XML. Secondly, because "*for the first time archivists have been offered a data structure standard that accommodates a hierarchical structure for the presentation of a variety of descriptions*" [Haworth, 2001]. Lastly, EAD enables archivists to be standard compliant and software independent.

In order to meet the expected requirements EAD reflects and emphasizes the hierarchical nature of ISAD(G) [Pitti, 1999]. EAD fully enables the expression of multiple description levels central to most archival descriptions and reflects hierarchy levels present in the resources being described. EAD cannot be considered a one-to-one ISAD(G) implementation, although it does respect ISAD(G) principles and is useful for representing archival hierarchical structure. EAD is composed of three high-level components: `<eadheader>`, `<frontmatter>`, and `<archdesc>`.

The `<eadheader>` contains metadata about the archive descriptions and includes information about them such as title, author, and date of creation. The `<frontmatter>` supplies publishing information and is an optional element, while the `<archdesc>` contains the archival description itself and constitutes the core of EAD. The `<archdesc>` may include many high-level sub-elements, most of which are repeatable. The most important element is the `<did>` or descriptive identification which describes the collection as a whole. The `<did>` element is composed of numerous sub-elements that are intended for brief, clearly designated statements of information and they are available at every level of description. Finally, the `<archdesc>` contains an element that facilitates a detailed analysis of the components of a fonds, the `<dsc>` or description subordinate components. The `<dsc>` contains a repeatable recursive element, called `<c>` or component. A component may be an easily recognizable archival entity such as series, subseries or items. Components not only are nested under the `<archdesc>` element, they are also usually nested inside one another. Components usually are indicated with `<cN>` tag, where $N \in \{01, 02, \ldots, 12\}$.

EAD reflects the archival structure and holds relations between entities in an archive. In addition, EAD encourages archivists to use collective and multilevel description, and because of its flexible structure and broad applicability, it has been embraced by many repositories [Kiesling, 2001].

On the other hand, EAD allows for several degrees of freedom in tagging practice, which may turn out to be problematic in the automatic processing of EAD files, since it is difficult to know in advance how an institution will use the hierarchical elements. The EAD permissive data model may undermine the very interoperability it is intended to foster. Indeed, it has been underlined that only EAD files meeting stringent best practice guidelines are shareable and searchable [Prom et al., 2007]. Moreover, there is also a second relevant problem related to the

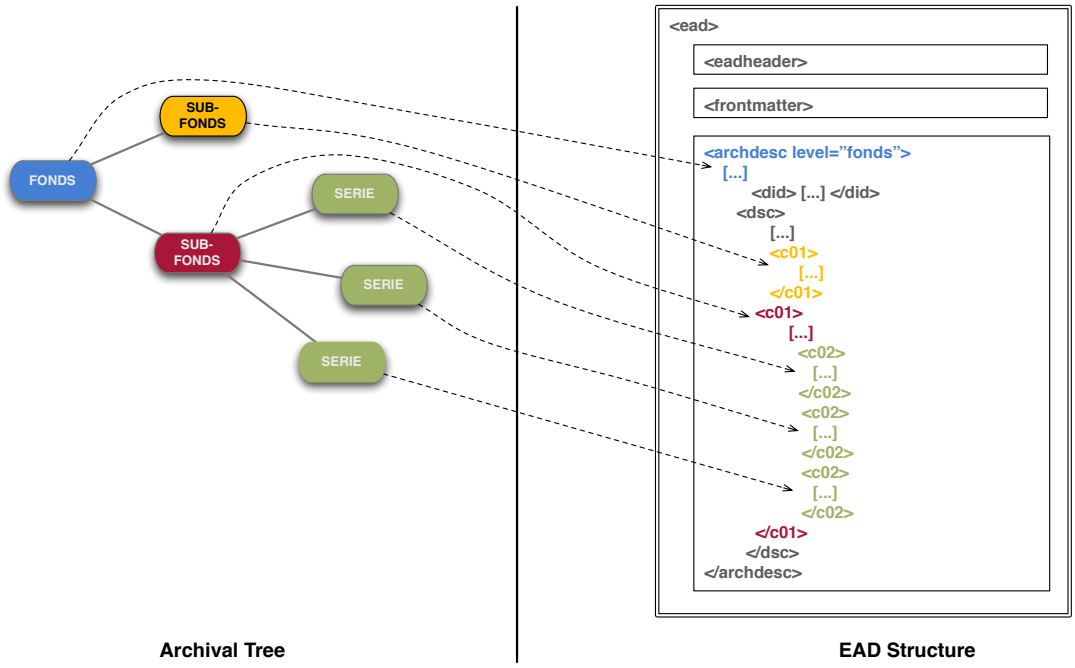


Figure 5.3: How an archive represented as a tree is mapped into an EAD XML file.

level of material that is being described. Unfortunately, the EAD schema rarely requires a standardized description of the level of the materials being described, since the `<level>` attribute is required only in the `<archdesc>` tag, while it is optional in `<cN>` components and in very few EAD files this possibility is used, as pointed out by [Prom, 2002]. As a consequence, the level of description of the lower components in the hierarchy needs to be inferred by navigating the upper components, maybe up to the `<archdesc>`, where the presence of the `<level>` attribute is mandatory. Therefore, access to individual items might be difficult without taking into consideration the whole hierarchy.

We highlight this fact in Figure 5.3 where we present the structure of an EAD file. In this example we can see the top-level components `<eadheader>` and `<archdesc>` and the hierarchical part represented by the `<dsc>` component; the `<level>` attribute is specified only in the `<archdesc>` component. Therefore, the archival levels described by the components of the `<dsc>` can be inferred only by navigating the whole hierarchy.

Moreover, sharing and searching archival description might be made difficult by the typical size of EAD files with a very deep hierarchical structure. Indeed, each EAD file is a hierarchical description of a whole collection of items rather than the description of an individual item [Shreeves et al., 2003]. On the other hand, users are often interested in the information described at the item level, which is typically buried very deeply in the hierarchy and might be difficult to reach.

Open Archive Initiative Protocol for Metadata Harvesting

The mission of the Open Archive Initiative (OAI)[15] has been to develop and promote interoperability standards that facilitate the efficient dissemination of digital library content. The Open Archive Initiative Protocol for Metadata Harvesting (OAI-PMH)[16] is a protocol which is a *de-facto* standard for the exchange of metadata between digital libraries in a distributed environment. OAI-PMH is designed to be open from an architectural point-of-view and a low-barrier mechanism for repository interoperability. It is non-intrusive, flexible, lightweight and easy to integrate into existing systems; these characteristics encouraged the adoption of OAI-PMH in almost all the applications and systems dealing with the exchange of metadata.

OAI-PMH is both application- and platform-independent; it is based on Web standards such as *HyperText Transfer Protocol (HTTP)* and XML and on two main components – *Data Provider* and *Service Provider*. Data Providers are repositories that export records in response to requests from a software service called harvester. On the other hand, Service Providers are those services that harvest records form Data Providers and provide services built on top of aggregated harvested metadata. OAI-PMH does not define the services to be offered by a service provider; that definition is left to the implementers of the Service Provider. The Service Provider harvests the metadata from the Data Providers – i.e. repositories – exploiting six *OAI-PMH verbs* based on HTTP requests: `identify` which is used to request a description of a Data Provider, `ListMetadataFormat` which lists the metadata formats supported by the Data Provider, `ListSets` which lists the sets defined by the data provider, `ListIdentifiers` which lists the unique identifiers associated with the metadata exposed by the Data Provider, `ListRecords` which lists all the metadata exposed by a Data Provider, and `GetRecord` which requests a specific metadata to a Data Provider. A Data Provider answers OAI-PMH requests with XML-encoded metadata records instance documents; the basic functioning of OAI-PMH is graphically represented in Figure 5.4. OAI-PMH does not impose any metadata format and thus the Data Providers are free to choose their own format or formats; although repositories are strongly encouraged to expose richer, possibly community-specific metadata formats, there is no requirement to do so. On the other hand, [Van de Sompel et al., 2002a] states that "*repository implementers should consider exporting the Dublin Core the first and most important step toward OAI-PMH interoperability. The addition of facilities to export other formats may then be added at a later date*". For this reason it is a common understanding that Dublin core is the minimum requirement for OAI-PMH, but the linkage between Dublin Core and OAI-PMH has been over-emphasized at the expense of the utility of OAI-PMH for dissemination of richer, and perhaps more useful, structured data. Currently, Dublin Core is not a requirement for the protocol but a recommendation.

The protocol defines two kinds of harvesting procedures: *incremental and selective harvesting*. Incremental harvesting permits users to query a Data Provider and ask it to return just the new, changed or deleted records from a certain date or between two dates. Selective har-

[15] `http://www.openarchives.org/`
[16] `http://www.openarchives.org/pmh/`

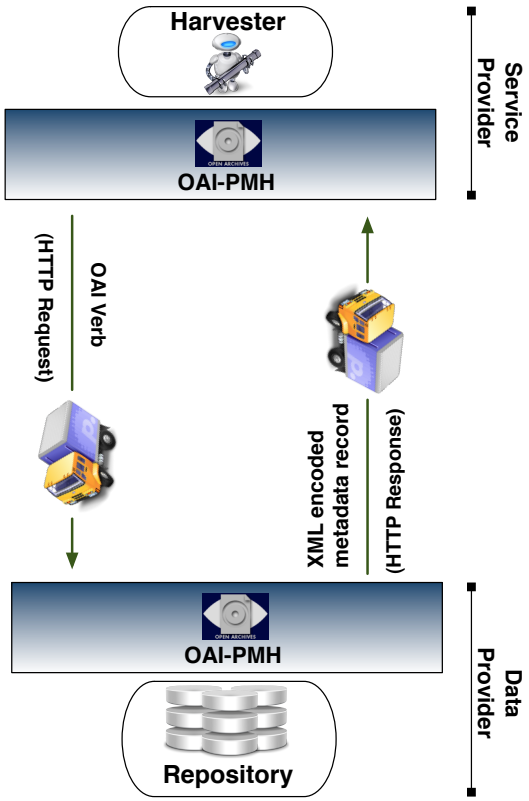
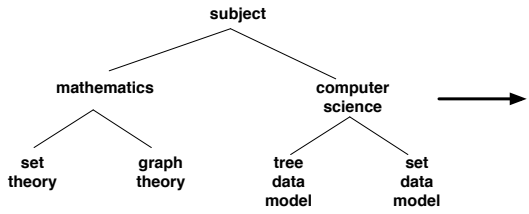


Figure 5.4: Basic functioning of OAI-PMH.

vesting is based on the concept of *OAI-set*, which enables logical data partitioning by defining groups of records. Selective harvesting is the procedure that permits the harvesting only of metadata owned by a specified OAI-Set. [Van de Sompel et al., 2003] states that in OAI-PMH a set is defined by three components: `setSpec` which is a mandatory and unique identifier for the set within the repository, `setName` which is a mandatory short human-readable string naming the set, and `setDesc` which may hold community-specific XML-encoded data about the set.

OAI set organization may be flat or hierarchical, where hierarchy is expressed in `setSpec` field by the use of a colon [:] separated list indicating the path from the root of the set hierarchy to the respective node. For example if we define an OAI-Set the `setSpec` of which is *"A"*, its sub-set "B" would have *"A:B"* as `setSpec`. In this case "B" is a proper sub-set of "A": $B \subset A$. Harvesting from a set which has sub-sets will cause the repository to return metadata in the specified set and recursively to return metadata from all the sub-sets. In our example, if we harvest set A, we also obtain the items in sub-set B [Van de Sompel et al., 2002b]. In Figure 5.5 we can see a hierarchical organization of OAI-Sets representing a library classification divided by subject.

It is worthwhile for the rest of the work to underline that a metadata encoded in XML is exchanged by OAI-PMH as an OAI record. An OAI record is metadata expressed in a single format and it is returned in an XML-encoded byte stream in response to an OAI-PMH request



SetName	SetSpec
subject	subject
mathematics	subject:mathematics
set theory	subject:mathematics:set_theory
graph theory	subject:mathematics:graph_theory
computer science	subject:computer_science
tree data model	subject:computer_science:tree_data_model
set data model	subject:computer_science:set_data_model

Hierarchical Classification of Subjects **Hierarchical Organization of OAI sets**

Figure 5.5: Hierarchical organization of OAI-Sets.

for metadata from an item. Such a record is composed of three main parts: *header*, *metadata* and *about*. The header part contains the basic information needed by OAI-PMH for its functioning which are the `unique identifier` identifying the item in the repository, the `datestamp` indicating the date and time of creation, update or deletion of the item in the repository, `setspec` specification which is a repeatable field reporting the sets at which the item belongs, and an optional `status` attribute with a value of deleted indicates the withdrawal of availability of the specified metadata format for the item. The attribute `setSpec` in the header of the record is particularly important because it permits us to infer to which sets a specific record belongs; this information is exploited by selective harvesting to retrieve all the records belonging to specific sets. Indeed, when a repository defines a set organization it must include set membership information in the headers of items returned to the harvester requests.

The *metadata* part of the record is pretty clear and it contains the metadata – expressed in whichever format – describing the corresponding item in the repository. The *about* part contains optional information about the *provenance* of the item and *right statements* if any.

Open Archive Initiative - Object Re-use and Exchange

The OAI-ORE defines a machine-readable and standard mechanism for defining aggregations of resources on the Web. By means of OAI-ORE we can identify a bunch of resources related to each other as a single entity enabling the access and exchange of them at an aggregation level of granularity. These aggregations are referred to by the OAI as "*compound objects*". Compound units are aggregations of distinct information units that, when combined, form a logical whole. Some examples [Van de Sompel and Lagoze, 2007] of these are a digitized book that is an aggregation of chapters, where each chapter is an aggregation of scanned pages, and a scholarly publication that is an aggregation of text and supporting materials such as datasets, software tools, and video recordings of an experiment. The OAI-ORE aims to "*develop mechanisms for representing and referencing compound information units in a machine-readable manner that is independent of both the actual content of the information unit and nature of the re-using application*" [Van de Sompel and Lagoze, 2007]. Since its first development in 2006 the goals of OAI-ORE are to:

- Facilitate discovery of compound objects.

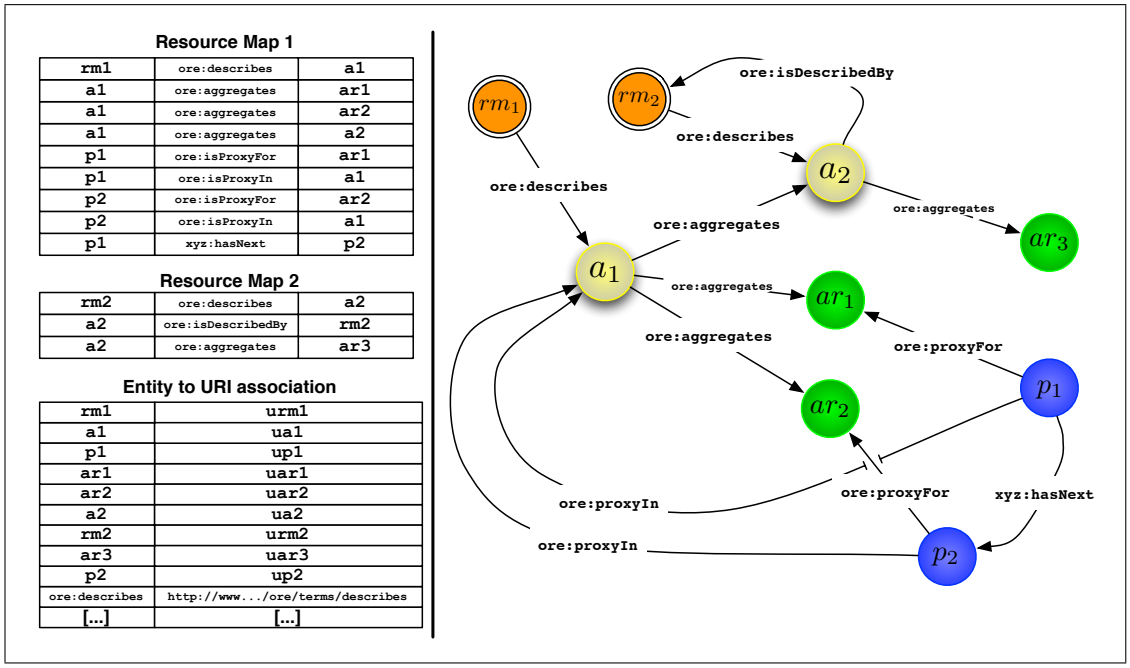
- Reference (link to) compound objects (and parts thereof).
- Obtain a variety of disseminations of compound objects.
- Aggregate and disaggregate compound objects.
- Enable processing by automated agents.

OAI-ORE represents an evolution of the OAI mission: from a *repository-centric* focus and a conceptualization of content as stored in repositories (i.e. OAI-PMH), which has characterized most digital library work, to a *resource-centric* focus in which machines act as service points to content independent of location [Lagoze et al., 2008a]. OAI-ORE heavily relies on the Web architecture and in particular exploits several concepts taken from the *Semantic Web*.

The OAI-ORE data model is based on three main kinds of resources: *Aggregation*, *Aggregated Resources* and *Resource Map*. Each resource in the OAI-ORE data model is identified by a *Uniform Resource Identifier (URI)*; the OAI-ORE specification [Lagoze et al., 2008b] uses URI-A to denote the URI of an Aggregation, URI-AR for an Aggregated Resource and URI-RM for a Resource Map. An Aggregation is defined as a resource representing a logical collection of other resources. An Aggregation is a logical construct and thus it has no representation; it is described by a Resource Map which can be seen as a materialization of the Aggregation. A Resource Map must describe a single Aggregation and must enumerate the constituent Aggregated Resources; a resource is an "Aggregated Resource" into an Aggregation only if it is asserted in a Resource Map.

The OAI-ORE data model can be represented throughout an RDF graph[17]. An RDF graph is defined by a set of triples (s, p, o) expressing the relationship defined by a predicate p between a subject s and an object o; each value in a triple is represented by a URI and URI are also used to name the relationships between s and o. Every URI referencing an Aggregation, an Aggregated Resource or a Resource Map can be used as s or o in a triple of the RDF graph representing the OAI-ORE model. Furthermore, OAI-ORE defines an extensible controlled vocabulary that associates each predicate with a URI. A Resource Map describes an Aggregation through a set of RDF triples; on the left-hand side of Figure 5.6 we can see the set of triples constituting two Resource Maps (rm_1 and rm_2) and materializing two Aggregations (a_1 and a_2). This triple states that the Resource Map rm_i identified by urm_i describes the Aggregation a_i identified by ua_i. Furthermore, a Resource Map must express minimal metadata properties indicating the authority defining the Resource Map, its last timestamp and, optionally, right information about the use of the Resource Map. OAI-ORE comes with another two important features: *Proxy* and *Nested Aggregations*. A Proxy is a resource that indicates an Aggregated Resource in the context of a specific Aggregation; a Proxy is associated with an Aggregated Resource via an assertion in a Resource Map describing the Aggregation that is the context of the Proxy [Lagoze et al., 2008b]. Proxies allow us to define relationships between Aggregated

[17] `http://www.w3.org/RDF/`



Resource Map 1

rm1	ore:describes	a1
a1	ore:aggregates	ar1
a1	ore:aggregates	ar2
a1	ore:aggregates	a2
p1	ore:isProxyFor	ar1
p1	ore:isProxyIn	a1
p2	ore:isProxyFor	ar2
p2	ore:isProxyIn	a1
p1	xyz:hasNext	p2

Resource Map 2

rm2	ore:describes	a2
a2	ore:isDescribedBy	rm2
a2	ore:aggregates	ar3

Entity to URI association

rm1	urm1
a1	ua1
p1	up1
ar1	uar1
ar2	uar2
a2	ua2
rm2	urm2
ar3	uar3
p2	up2
ore:describes	http://www.../ore/terms/describes
[...]	[...]

Figure 5.6: RDF triples and their graph representation.

Resources; for instance, if an Aggregation contains the chapters of a book as independent Aggregated Resources, via Proxies we can define a precedence order between the chapters. In Figure 5.6 we can see two proxies p_1 and p_2 defining an order of precedence between the Aggregated Resources ar_1 and ar_2 in the context of Aggregation A_1.

The *Nested Aggregations* feature enables the definion of Aggregations of Aggregations; this is consistent in the OAI-ORE data model because an Aggregation is a Resource which can also be seen as an Aggregated Resource of another Aggregation. In Figure 5.6 we show two nested Aggregations a_1 and a_2.

Although OAI-ORE is a relatively young specification, it has been becoming a standard reference in the context of digital libraries and its use is widespread in many systems and applications that deal with aggregations of digital objects. As with OAI-PMH, the use of OAI-ORE has been firstly adopted for the management, access, and curation of scholarly publications [Cheung et al., 2008; Mikeal et al., 2009] and now it is spreading into other digital library related fields such as the management and representation of scientific data [Brooking et al., 2009; Pepe et al., 2010].

Chapter 6

The NESTOR Model

In this chapter we define the NESTOR model which represents the central component of the NESTOR Framework and thus one of the main contributions of this work.

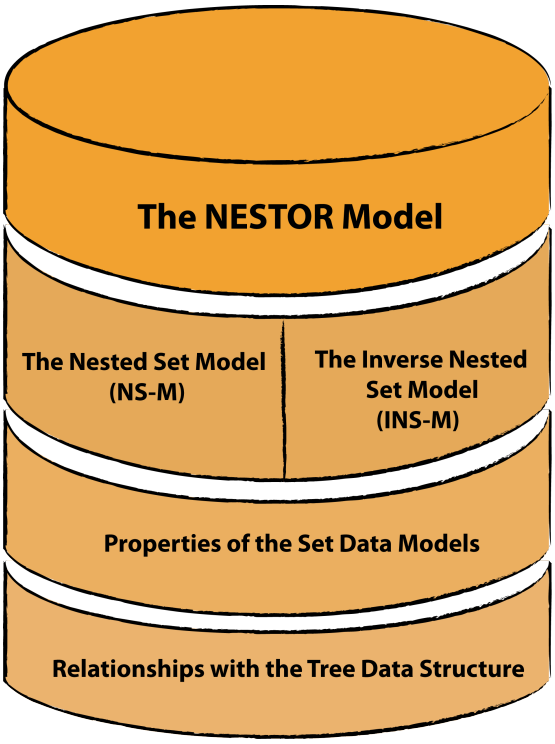


Figure 6.1: The main components of the NESTOR Model.

In Figure 6.1 we can see the main parts of the NESTOR Model which represents also the skeleton of the organization of this chapter.

First-of-all we define the NS-M and the INS-M. In the second part we present the set-theoretical properties of both the NS-M and the INS-M and the functions to go from a model to the other and vice versa. In the third part we define the formal relationships between both the set data models and the tree, how to go from the NS-M and the INS-M to a tree, and how the properties of the tree are defined in the set data models. We show that all the characteristics of

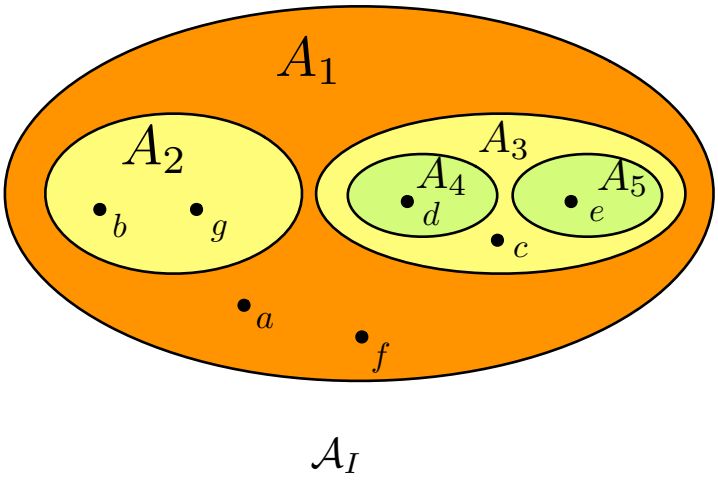


Figure 6.2: A sample Nested Set Family represented by means of an Euler-Venn diagram.

the tree are retained by the set data models. We define two extensions of the set data models that allow us to compare these models with the extensions of the tree – i.e. ordered trees. Moreover, we define a further extension of the set data models that does not find a correspondent one in the tree – i.e. the set data models defined as families of partially ordered sets which allow us to define and handle the structure of the elements belonging to the sets.

Lastly, we define and prove the metric properties of the set data models comparing them with the metrics defined for the tree.

6.1 The Set Data Models

The NESTOR Model defines two set data models that allow us to represent and manage hierarchical structures. These set data models – i.e. *Nested Set Model (NS-M)* and *Inverse Nested Set Model (INS-M)* – are independent one from the other and, at the same time, they are related by formally defined functions that permit us to go from a model to the other [Agosti et al., 2010b, 2009a; Ferro and Silvello, 2009b; Silvello, 2008].

Definition 6.1.
Let A be a set and let $\mathcal{A}_I$ be a family of sets. Then $\mathcal{A}_I$ is a ***Nested Set Family*** *if:*

$$A \in \mathcal{A}_I, \tag{6.1}$$

$$\emptyset \notin \mathcal{A}_I, \tag{6.2}$$

$$\forall A_h, A_k \in \mathcal{A}_I, h \neq k \mid A_h \cap A_k \neq \emptyset \Rightarrow A_h \subset A_k \vee A_k \subset A_h. \tag{6.3}$$

Thus, we define a *Nested Set Family* (NS-F) as a family of sets where three conditions must hold. The first condition (6.1) states that set A which contains all the sets in the family must

belong to the NS-F. The second condition states that the empty-set does not belong to the NS-F and the last condition (6.3) states that the intersection of every couple of distinct sets in the NS-F is not the empty-set only if one set is a proper subset of the other one [Anderson and Hall, 1963; Halmos, 1960].

The NS-F are represented by means of Eulero-Venn diagrams as we can see in Figure 6.2 which represents a sample NS-F composed by five nested sets: $\mathcal{A}_I = \{A_1, A_2, A_3, A_4, A_5\}$. We can see that A_1 is the common superset of all the sets in $\mathcal{A}_I$ and thus Condition 6.1 is respected, there is no empty set in the family (Condition 6.2) and all the sets are either disjoints or one a proper subset of the other which is required by Condition 6.3.

In the same way we can define the Inverse Nested Set Model (INS-M):

Definition 6.2.
Let A be a set and let $\mathcal{A}_I$ be a family. Then $\mathcal{A}_I$ is an ***Inverse Nested Set Family*** *if:*

$$\emptyset \notin \mathcal{A}_I, \tag{6.4}$$

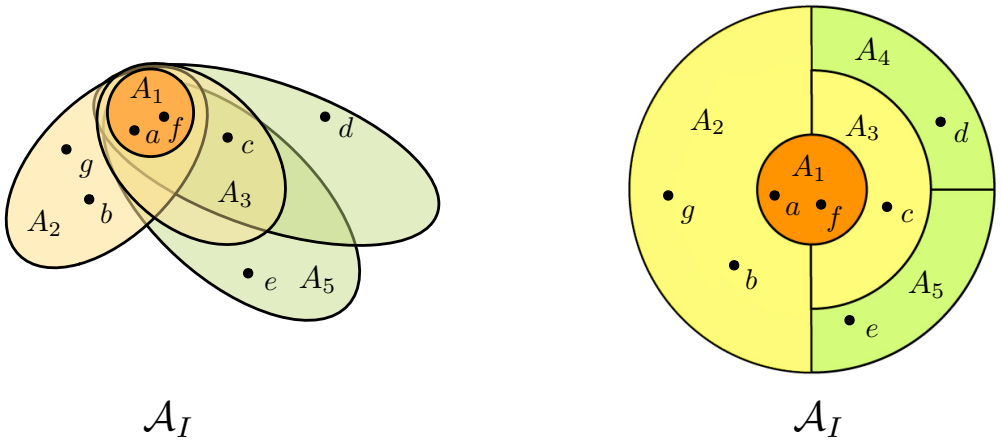
$$\forall \mathcal{A}_J \subseteq \mathcal{A}_I \Rightarrow \bigcap_{j \in J} A_j \in \mathcal{A}_I. \tag{6.5}$$

$$\begin{aligned} &\forall \mathcal{A}_J \subseteq \mathcal{A}_I \\ &\Rightarrow \exists A_k \in \mathcal{A}_J \mid \forall A_h \in \mathcal{A}_J, A_h \subseteq A_k \\ &\Rightarrow \forall A_h, A_g \in \mathcal{A}_J, A_h \subseteq A_g \vee A_g \subseteq A_h. \end{aligned} \tag{6.6}$$

Thus, we define an Inverse Nested Set Family (INS-F) as a family where three conditions must hold. The first condition (6.4) states that the empty-set does not belong to the INS-F. The second condition (6.5) states that the intersection of every subfamily of the INS-F belongs to the INS-F itself. Condition 6.6 states that every subfamily of a INS-F can be a topped family only if it is linearly ordered.

Unlike the NS-M, the representation of the INS-M by means of the Euler-Venn diagrams is not very expressive and can be confusing for the reader [Agosti et al., 2009a] – see Figure 6.3a. We can represent in a straightforward way the INS-M by means of the "*DocBall representation*" [Crestani et al., 2004] – see Figure 6.3b. The DocBall representation is used in [Crestani et al., 2004] to depict the structural components of the documents and can be considered as the representation of a tree structure. We exploit the DocBall ability to show the structure of an object and to represent the "*inclusion order of one or more elements in another one*" [Vegas et al., 2007]. The DocBall is composed of a set of circular sectors arranged in concentric rings as shown in Figure 6.3b. In a DocBall each ring represents a level of the hierarchy with the center (level 0) representing the root. In a ring, the circular sectors represent the nodes in the corresponding level. We use the DocBall to represent the INS-M, thus for us each circular sector corresponds to a set.

In Figure 6.3b we can see a DocBall representing the same INS-M represented by an Euler-Venn diagram in Figure 6.3a. The set A_1 which is subset of all the other sets in the INS-M is



(a) Euler-Venn Representation of an INS-M **(b) DocBall Representation of an INS-M**

Figure 6.3: A sample Inverse Nested Set Family represented by means of an Euler-venn diagram and a DocBall.

represented by the inner ring at level 0 of the DocBall; at level 1 we find the direct supersets of A_1 which are A_2 and A_3; both these sets are represented as circular sectors comprising the inner circle representing A_1. With this representation a subset is presented in a ring within the set including it. Indeed, we can see that set A_1 is included by all the other sets. If the intersection of two or more sets is empty then these sets have no common circular sector in the inner rings of the DocBall. For instance, we can see that the circular sectors A_2 and A_5 have in common only A_1, indeed $A_2 \cap A_5 = A_1$; instead, A_4 and A_5 have in common the sectors A_3 and A_1, thus $A_4 \cap A_5 = \{A_3, A_1\}$.

So, we represent the INS-F by means of the DocBall[1] as we can see in Figure 6.3 which represents a sample INS-F $\mathcal{A}_I$ composed by five sets. As we can see there is no empty set in the family thus Condition 6.4 is respected; the intersection of every two sets is a set in the family (Condition 6.5) and there is not a set which is a common superset of all the other sets, thus the family is not topped, then also Condition 6.6 is respected.

6.2 Properties of the Set Data Models

In this section we present the basic set-theoretical properties of the two set data models: NS-M and INS-M. First-of-all we present the properties of the NS-M and then we symmetrically present the properties of the INS-M. Most of the set-theoretic properties of the set data models are straightforwardly derived from the very definition of the models or from the set-theory basics.

[1]The graphical representation based on the DocBall exploited an idea that was conceived to represent the hierarchical structure of a Web page [Vegas et al., 2003] but it does not have any relations with the model underlying it.

6.2.1 Properties of Nested Set Model

The following proposition describes the fundamental properties of the NS-M showing how the sets in a NS-F behave under the three main set-theoretical operations: union, intersection and difference.

Proposition 6.1. *Let $\mathcal{A}_I$ be a NS-F, $A \in \mathcal{A}_I$ be the set such that $\forall A_j \in \mathcal{A}_I, A_j \subseteq A$ and X be a set. Then $\forall A_j, A_k \in \mathcal{A}_I$, $j \neq k$:*

- ***Union:***

$$(A_j \subset A_k) \Rightarrow (A_j \cup A_k = A_k) \tag{6.7}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow A_j \cup A_k = X \subseteq A \tag{6.8}$$

- ***Intersection:***

$$(A_j \subset A_k) \Rightarrow (A_j \cap A_k = A_j) \tag{6.9}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow A_j \cap A_k = \emptyset \notin \mathcal{A}_I \tag{6.10}$$

- ***Difference:***

$$A_k \subset A_j \Rightarrow A_j \setminus A_k = X \subset A \tag{6.11}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow (A_j \setminus A_k = A_j) \wedge (A_k \setminus A_j = A_k) \tag{6.12}$$

Proof:
Properties 6.7, 6.9 and 6.12 are straightforwardly derived from set-theory and they do not need to be proved. Property 6.10 comes straightforwardly from condition 6.3 of Definition 6.1.

***Property 6.8**: Ab absurdo suppose that $A_j \cup A_j = X \not\subseteq A \Rightarrow \exists a \in X \mid a \notin A \Rightarrow a \in A_j \vee a \in A_k \Rightarrow A_j \not\subseteq A \vee A_k \not\subseteq A$.*

***Property 6.11**: Ab absurdo suppose that $A_j \setminus A_k = X \not\subseteq A \Rightarrow \exists a \in X \mid a \notin A \Rightarrow a \in A_j \Rightarrow A_j \not\subseteq A$.* □

We present a couple of examples to illustrate these properties.

Example 6.1. *Let $\mathcal{A}_I = \{A_r, A_k, A_j\}$ be a NS-F, where $A_r = \{a, b, c, d, e\}$, $A_j = \{c, d, e\}$ and $A_k = \{e\}$; the NS-F is represented in Figure 6.4 (a).*

In this example $A_k \subset A_j$. Then, $A_j \cup A_k = \{c, d, e\} = A_j$, $A_j \cap A_k = \{e\} = A_k$ and $X = A_j \setminus A_k = \{c, d\}$, we can see that $X \notin \mathcal{A}_I$ but $X \subset A_r$.

Example 6.2. *Let $\mathcal{A}_I = \{A_r, A_k, A_j\}$ be a NS-F, where $A_r = \{a, b, c, d, e\}$, $A_j = \{c, d\}$ and $A_k = \{e\}$; the NS-F is represented in Figure 6.4 (b).*

In this example $A_k \not\subseteq A_j \wedge A_j \not\subseteq A_k$. Then, $A_j \cup A_k = \{c, d, e\} = X \notin \mathcal{A}_I$ but $X \subset A_r$, $A_j \cap A_k = \emptyset \notin \mathcal{A}_I$ and $A_j \setminus A_k = \{c, d\} = A_j$.

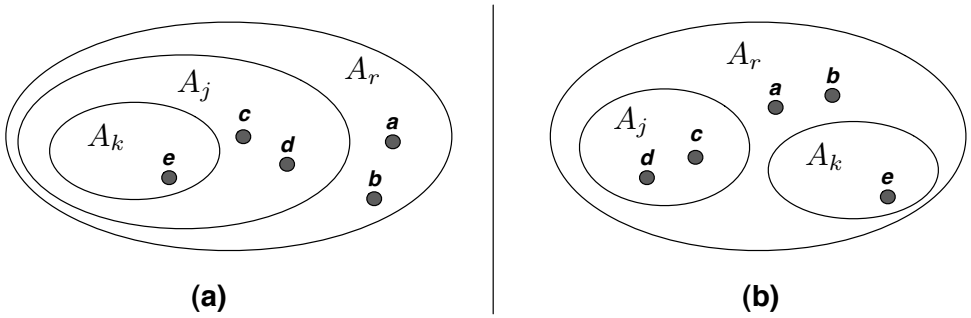


Figure 6.4: The graphical representation through an Euler-Venn diagrams of the NS-F used in Example 6.1 (a) and Example 6.2 (b).

The properties 6.8 and 6.11 require a further explanation for what is concerned with the set X; it is not formally specified whether or not this set belongs to the NS-F. In order to understand the role of this set, we need the following corollary.

Corollary 6.2. *Let $\mathcal{A}_I$ be a NS-F, $A_j, A_k, A_t \in \mathcal{A}_I$, $j \neq k \neq t$ be three sets, where $A_j \nsubseteq A_k \wedge A_k \nsubseteq A_j$. If:*

$$A_j \cup A_k = A_t \Rightarrow \mathcal{D}^-(A_j) = \mathcal{D}^-(A_k) = A_t \tag{6.13}$$

Proof:
Ab absurdo suppose that $(A_j, A_k, A_t \in \mathcal{A}_I) \wedge (A_j \cup A_k = A_t) \Rightarrow \mathcal{D}^-(A_j) \neq \mathcal{D}^-(A_k) \neq A_t$. This means that $\mathcal{D}^-(A_j) = A_g \in \mathcal{A}_I$ and that $\mathcal{D}^-(A_k) = A_f \in \mathcal{A}_I, g \neq f \neq t \Rightarrow A_j \subset A_g \subset A_t \wedge A_k \subset A_f \subset A_t$. $A_j \cup A_k = A_t \Rightarrow A_t \subset (A_g \cup A_k) \Rightarrow (A_k \nsubseteq A_t) \vee (A_g \nsubseteq A_t)$. At the same time $A_j \cup A_k = A_t \Rightarrow A_t \subset (A_j \cup A_f) \Rightarrow (A_j \nsubseteq A_t) \vee (A_f \nsubseteq A_t)$. This leads to the conclusion that $A_t \notin \mathcal{A}_I \vee A_j \cup A_k \neq A_t$. □

Corollary 6.2 is important because it allows the existence of *partitioned sets* in a NS-F. Indeed, if $A_j \cup A_k = A_t$, the sets A_j and A_k form a partition of the set A_t in the family $\mathcal{A}_I$. We can also say that the elements of A_k and A_j **cover** the set A_t. A direct consequence is that if the union of two sets in a NS-F is a set belonging to the family itself, then this set is the direct superset of these two sets; otherwise the set generated from the union does not belong to the family.

Example 6.3. *In Figure 6.5 we can see a graphical representation of the case proved in the Corollary 6.2. Furthermore, this corollary explains the role of the set X in Proposition 6.1 and by means of Property 6.13 we can explain Property 6.11 of Proposition 6.1: $A_k \subset A_j \Rightarrow A_j \setminus A_k = X \subset A$. We can say that $X \in \mathcal{A}_I$ if $\mathcal{D}^-(X) = \mathcal{D}^-(A_k) = A_j$ meaning that the set X in the NS-F $\mathcal{A}_I$ only if A_j is a set partitioned by A_k and X itself. If we consider the NS-F in Figure 6.5 (a) we can see that $A_k \cup A_j \neq A_t$; indeed, $A_t = \{a, b, c\}$ and $A_k \cup A_j = \{b, c\} \notin \mathcal{A}_I$. In*

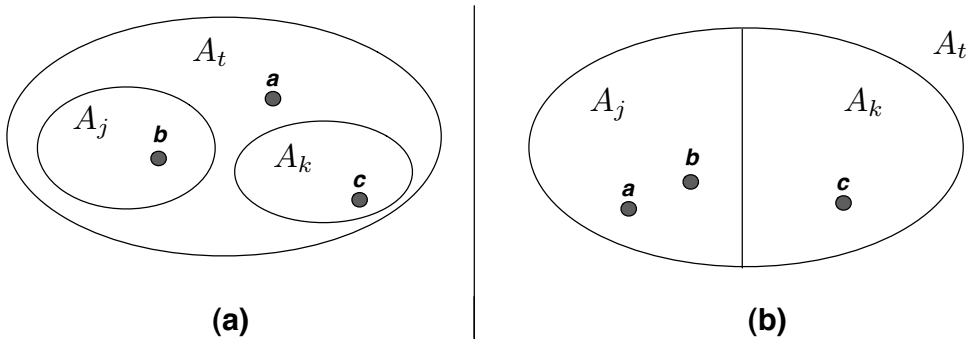


Figure 6.5: (a) The set A_t is not partitioned by A_j and A_k. (b) The set A_t is partitioned by A_j and A_k.

Figure 6.5 (b) the set A_t is partitioned and $A_t = \{a, b, c\} = A_j \cup A_k \in \mathcal{A}_I$.

We have to consider also the symmetric difference between two sets; in a NS-F the meaning of this operation can be inferred from the properties described by Proposition 6.1 which are used to prove the following proposition.

Proposition 6.3. *Let $\mathcal{A}_I$ be a NS-F, $A_t \in \mathcal{A}_I \mid \forall A_w \in \mathcal{A}_I, A_w \subseteq A_t$ be a set in the family and X be a set. Then $\forall A_j, A_k \in \mathcal{A}_I$, $j \neq k$:*

$$A_j \triangle A_k = X \subset A_t \tag{6.14}$$

Proof:
We have to prove two cases; in the first case: $A_j \subset A_k$. In the second case: $(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j)$.

Let us prove the first case. From properties 6.7, 6.9 and 6.11 of Proposition 6.1 we know that $A_j \triangle A_k = (A_j \cup A_k) \setminus (A_j \cap A_k) = A_k \setminus A_j = X \subset A$.

Now, let us prove the second case. From properties 6.8, 6.10 and 6.12 of Proposition 6.1 we know that $A_j \triangle A_k = (A_j \cup A_k) \setminus (A_j \cap A_k) = X \setminus \emptyset = X \subset A$. □

6.2.2 Properties of Inverse Nested Set Model

The following proposition describes the fundamental properties of the INS-M showing how the sets in an INS-F behave under the three main set-theoretical operations: union, intersection and difference.

Proposition 6.4. *Let $\mathcal{A}_I$ be a INS-F and let X be a set. Then $\forall A_j, A_k \in \mathcal{A}_I$, $j \neq k$:*

- *union:*

$$(A_j \subset A_k) \Rightarrow (A_j \cup A_k = A_k) \tag{6.15}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow A_j \cup A_k = X \notin \mathcal{A}_I \tag{6.16}$$

- *intersection:*

$$(A_j \subset A_k) \Rightarrow (A_j \cap A_k = A_j) \tag{6.17}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow \exists A_t \in \mathcal{A}_I \mid A_j \cap A_k = A_t \tag{6.18}$$

- *difference*

$$A_k \subset A_j \Rightarrow A_j \setminus A_k = X \notin \mathcal{A}_I \tag{6.19}$$

$$(A_j \not\subseteq A_k) \wedge (A_k \not\subseteq A_j) \Rightarrow A_j \setminus A_k = A_k \setminus A_j = X \notin \mathcal{A}_I \tag{6.20}$$

Proof:
Properties 6.15 and 6.17 are straightforwardly derived from set-theory and they do not need to be proved. Property 6.18 comes straightforwardly from condition 6.5 of Definition 6.2.

***Property 6.16**: Ab absurdo suppose that $A_j \not\subseteq A_k \wedge A_k \not\subseteq A_j \Rightarrow A_j \cup A_k = X \in \mathcal{A}_I$. This means that $\exists \mathcal{B}_T = \{A_j, A_k, X\} \subset \mathcal{A}_I$ such that it is topped but not linearly ordered.*

***Property 6.19**: Ab absurdo suppose that $A_k \subset A_j \Rightarrow A_j \setminus A_k = X \in \mathcal{A}_I$. This means that $X \cap A_j = \emptyset \Rightarrow \exists \mathcal{B}_T = \{A_j, X\} \subset \mathcal{A}_I \mid \bigcap \mathcal{B}_T \notin \mathcal{A}_I$.*

Property 6.20 can be proved following exactly the same steps of the proof of property 6.19.□

In this case the role of the set X has been already explained in the proof of the proposition. We can see that in an INS-F we cannot find partitioned sets because they violate the condition 6.5 of Definition 6.2: let $\mathcal{A}_I$ be an INS-F, $A_j \in \mathcal{A}_I$ be a set and suppose that A_j is partitioned by the sets Y and Z, then $X \cap Z = \emptyset \notin \mathcal{A}_I$ thus, the intersection of two sets in the family does not belong to the family itself.

Example 6.4. *Let $\mathcal{A}_I = \{A_r, A_k, A_j\}$ be a INS-F, where $A_r = \{a, b\}$, $A_j = \{a, b, c, d\}$ and $A_k = \{a, b, c, d, e\}$; the INS-F is represented in Figure 6.6 (a).*

In this example $A_j \subset A_k$. Then, $A_j \cup A_k = \{a, b, c, d, e\} = A_k$, $A_j \cap A_k = \{a, b, c, d\} = A_j$ and $X = A_k \setminus A_j = \{e\} \notin \mathcal{A}_I$.

Example 6.5. *Let $\mathcal{A}_I = \{A_r, A_k, A_j\}$ be a INS-F, where $A_r = \{a, b\}$, $A_j = \{a, b, c, d\}$ and $A_k = \{a, b, e\}$; the INS-F is represented in Figure 6.6 (b).*

In this example $A_k \not\subseteq A_j \wedge A_j \not\subseteq A_k$. Then, $A_j \cup A_k = \{a, b, c, d, e\} = X \notin \mathcal{A}_I$, $A_j \cap A_k = \{a, b\} = A_r \in \mathcal{A}_I$ and $A_j \setminus A_k = \{c, d\} \notin \mathcal{A}_I$.

Proposition 6.5. *Let $\mathcal{A}_I$ be a INS-F and Y be a set. Then $\forall A_j, A_k \in \mathcal{A}_I$, $j \neq k$:*

$$A_j \triangle A_k = Y \notin \mathcal{A}_I \tag{6.21}$$

Proof:
We have to consider two cases; in the first case: $A_j \subset A_k$. In the second case: $A_j \not\subseteq A_k \wedge A_k \not\subseteq A_j$.

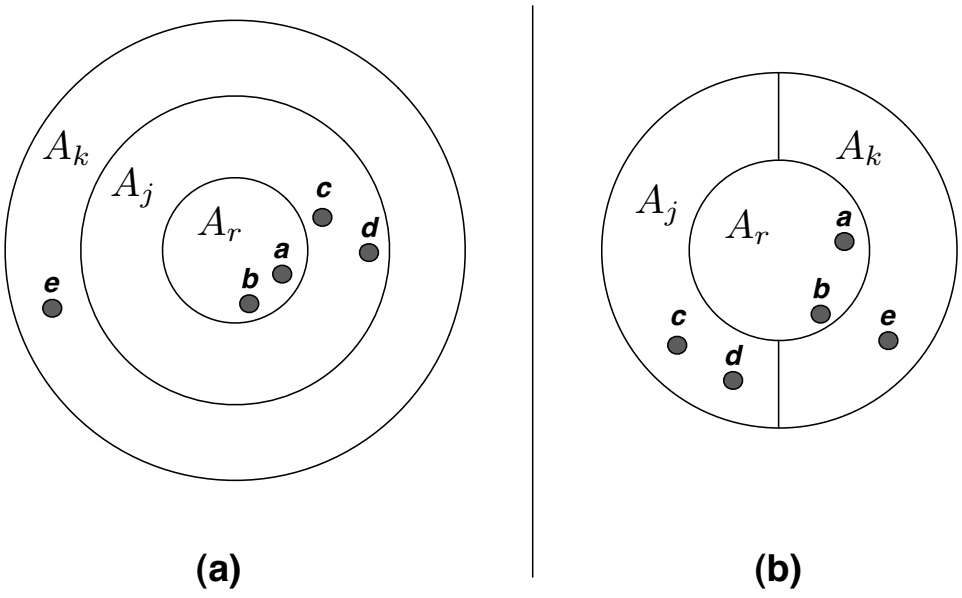


Figure 6.6: (a) A linearly ordered INS-F (b) An INS-F.

Let us prove the first case. From properties 6.15, 6.17 and 6.19 of Proposition 6.4 we know that $A_j \triangle A_k = (A_j \cup A_k) \setminus (A_j \cap A_k) = A_k \setminus A_j = X \notin \mathcal{A}_I$.

Let us prove the second case. From properties 6.16, 6.18 and 6.20 of Proposition 6.4 we know that $A_j \triangle A_k = (A_j \cup A_k) \setminus (A_j \cap A_k) = X \setminus A_t$ *where* $X \notin \mathcal{A}_I \wedge A_t \in \mathcal{A}_I$. *Then we define* $Y = X \setminus A_t$. *Ab absurdo suppose that* $Y \in \mathcal{A}_I \Rightarrow A_t \cap Y = \emptyset \Rightarrow \mathcal{A}_I$ *is not a INS-F.* □

6.3 Mapping Between the Set Data Models

We have defined the two set data models as families of sets that have to respect a bunch of conditions; in the following we present two functions ζ and ξ which allow us to map a family of sets into another family and we will show how these functions permit us to go back and forth from a NS-F to a INS-F and vice versa. Please note that in the following definitions we widely exploit the concept of collection of supersets (Definition 4.12, Section 4.1.4, Chapter 4).

Definition 6.3.
Let $\mathcal{A}_I$ *and* $\mathcal{B}_J$ *be two families of sets. We define* $\zeta : \mathcal{A}_I \rightarrow \mathcal{B}_J$ *to be a function such that for all* $A_k \in \mathcal{A}_I$ *there exists* $B_k \in \mathcal{B}_J$ *such that:*

$$B_k = \bigcup_{A_t \in \{A_k \cup \mathcal{S}^-(A_k)\}} (A_t \setminus \bigcup \mathcal{S}^+(A_t)) \tag{6.22}$$

For every set $A_k \in \mathcal{A}_I$, the ζ function takes into account all its supersets – i.e. $A_k \cup \mathcal{S}^-(A_k)$; for each one of these supersets (call them A_t) the ζ function retains all the elements that exclusively belong to A_t – i.e. the elements which are in A_t and do not belong to any other

subset of A_t – i.e. $A_t \setminus \bigcup \mathcal{S}^+(A_t)$. Then, the set $B_k = \zeta(A_k)$ contains the union of all the elements of all the considered A_t.

Definition 6.4.
Let $\mathcal{A}_I$ and $\mathcal{B}_J$ be two families of sets. We define $\xi : \mathcal{A}_I \to \mathcal{B}_J$ to be a function such that for all $A_k \in \mathcal{A}_I$ there exists $B_k \in \mathcal{B}_J$ such that:

$$B_k = \bigcup (A_k \cup \mathcal{S}^-(A_k)) \setminus \bigcup \mathcal{S}^+(A_k) \tag{6.23}$$

The ξ function maps every set $A_k \in \mathcal{A}_I$ into another set, call it $B_k \in \mathcal{B}_J$. B_k is defined by the union of all the elements belonging to A_k and to its supersets minus all the elements belonging to the subsets of A_k itself.

It is important to appreciate the difference between these two functions. In ζ a set A_k is mapped in a set B_k by taking the set A_k minus all the elements in its subsets; then this operation is repeated for each superset of A_k and the elements returned by these operations are contained by the set B_k.

In ξ a set A_k is mapped in a set B_k by taking the set A_k together with all its supersets and then subtracting all the elements belonging to the subsets of A_k.

The next theorem proves that if we apply the above defined function ζ to a NS-F we obtain an INS-F as output.

Theorem 6.6. *Let $\mathcal{A}_I$ be a NS-F then $\zeta(\mathcal{A}_I) = \mathcal{B}_J$ is an INS-F.*

Proof:
To prove that $\zeta(\mathcal{A}_I) = \mathcal{B}_J$ is an INS-F we have to verify if it satisfies the three conditions of Definition 6.2.
***Condition (6.4)**. We know that $\emptyset \notin \mathcal{A}_i$ thus $\forall A_k \in \mathcal{A}_i$, $A_k \neq \emptyset \Rightarrow \forall A_t \in \{A_k\} \cup \mathcal{S}^-(A_k), A_t \neq \emptyset$ and from the definition of NS-F we know that $|A_t| > |\bigcup \mathcal{S}^+(A_t)| \Rightarrow A_t \setminus \bigcup \mathcal{S}^+(A_t) \neq \emptyset$. Then $\forall B_k \in \mathcal{B}_J$, $B_k \neq \emptyset \Rightarrow \emptyset \notin \mathcal{B}_J$.*
***Condition (6.5)**. We have to prove that the intersection of every two sets $B_i, B_j \in \mathcal{B}_J$ is a set belonging to $\mathcal{B}_J$. Please note that by definition of NS-F $\exists A \in \mathcal{A}_I \mid \forall A_k \in \mathcal{A}_I, A_k \subseteq A$, then we can state that $\forall A_k \in \mathcal{A}_I$:*

(a) $\bigcup \mathcal{S}^+(A_k) \subseteq \bigcup \mathcal{S}^+(A)$.

(b) $\bigcup \mathcal{S}^-(A) = \emptyset$.

(c) $A \in \mathcal{A}_I$, $B \in \mathcal{B}_J \mid \zeta(A) = A \setminus \bigcup \mathcal{S}^+(A) = B \Rightarrow \forall A_k \in \mathcal{A}_I$, $A_k \subseteq A \Rightarrow \forall B_k \in \mathcal{B}_J$, $B \subseteq B_k$.

Then, we know that $\forall A_i, A_j \in \mathcal{A}_I$:

(d) $A_i \cap A_j \neq \emptyset \Rightarrow A_i \subset A_j \vee A_j \subset A_i$ *by Property 6.3 of Definition 6.1.*

(e) $B_i = \zeta(A_i) = \bigcup_{A_t \in \{A_i \cup \mathcal{S}^-(A_i)\}} (A_t \setminus \bigcup \mathcal{S}^-(A_t))$.

(f) $B_j = \zeta(A_j) = \bigcup_{A_t \in \{A_j \cup \mathcal{S}^-(A_j)\}}(A_t \setminus \bigcup \mathcal{S}^-(A_t))$.

Now, we can show that:

1. *If* $A_i \subset A_j \Rightarrow (\mathcal{S}^-(A_j) \subset \mathcal{S}^-(A_i)) \wedge (\mathcal{S}^+(A_i) \subset \mathcal{S}^+(A_j))$. *Then* $\zeta(A_j) \subset \zeta(A_i) = B_j \subset B_i \Rightarrow B_j \cap B_i = B_j \in \mathcal{B}_J$.

2. *If* $A_j \subset A_i \Rightarrow (\mathcal{S}^-(A_i) \subset \mathcal{S}^-(A_j)) \wedge (\mathcal{S}^+(A_j) \subset \mathcal{S}^+(A_i))$. *Then* $\zeta(A_i) \subset \zeta(A_j) = B_i \subset B_j \Rightarrow B_i \cap B_j = B_i \in \mathcal{B}_J$.

3. *If* $A_j \cap A_i = \emptyset \Rightarrow (A_j \not\subseteq A_i) \wedge (A_i \not\subseteq A_j)$. *We know that* $A_j \subset A \wedge A_i \subset A \Rightarrow \zeta(A) \subset \zeta(A_i) \wedge \zeta(A) \subset \zeta(A_j)$.
We know that $\exists\ A_k \in \mathcal{A}_I \mid \bigcup \mathcal{S}^-(A_j) \cap \bigcup \mathcal{S}^-(A_i) \subseteq A_k \subseteq A$ *and that* $\bigcup \mathcal{S}^+(A_j) \cap \bigcup \mathcal{S}^+(A_i) = \emptyset$.
Then $\forall A_i, A_j \in \mathcal{A}_I \mid A_i \cap A_j = \emptyset, \exists\ A_k \in \mathcal{A}_I \mid A_j \subset A_k \wedge A_i \subset A_k \Rightarrow \exists\ B_k \in \mathcal{B}_J \mid B_k \subset B_i \wedge B_k \subset B_j \Rightarrow \forall B_i, B_j \in \mathcal{B}_J, \exists\ B_k \in \mathcal{B}_J \mid B_i \cap B_j = B_k \in \mathcal{B}_J$.

Condition (6.6)*. Let* $\zeta(\mathcal{A}_I) = \mathcal{B}_J$ *and let us consider a* $C_k \subseteq \mathcal{B}_J$*; ab absurdo suppose that* C_k *is topped but not linearly ordered thus* $\exists C_i, C_j, C_h \in C_k \mid (C_i \cap C_j \neq \emptyset) \wedge (C_i \cup C_j \subseteq C_h) \wedge (C_i \not\subseteq C_j) \wedge (C_j \not\subseteq C_i)$*. We know that* $\zeta(A_i) = C_i, \zeta(A_j) = C_j$ *and* $\zeta(A_h) = C_h$*; if* $C_i \not\subseteq C_j \wedge C_j \not\subseteq C_i$ *then* $A_j \not\subseteq A_i \wedge A_i \not\subseteq A_j$*. At the same time* $C_i \subseteq C_h \wedge C_j \subseteq C_h \Rightarrow A_h \subseteq A_i \wedge A_h \subseteq A_j \Rightarrow A_i \cap A_j = A_h$*. Then* $\mathcal{A}_I$ *is not a NS-F.* □

Let us see an example showing how the ζ function can be applied to the sample NS-F showed in Figure 6.2.

Example 6.6. *Let* $\mathcal{A}_I$ *be a NS-F and let* $\mathcal{A}_I = \{A_1, A_2, A_3, A_4, A_5\}$ *where* $A_1 = \{a, b, c, d, e, f, g\}$, $A_2 = \{b, g\}$, $A_3 = \{c, d, e\}$, $A_4 = \{d\}$ *and* $A_5 = \{e\}$. *Then* $\zeta(\mathcal{A}_I) = \mathcal{B}_J = \{B_1, B_2, B_3, B_4, B_5\}$, *where:*

$\zeta(A_1) = B_1 = \bigcup_{A_t \in \{A_1 \cup \mathcal{S}^-(A_1)\}}(A_t \setminus \bigcup \mathcal{S}^+(A_t)) = A_1 \setminus \bigcup\{A_2, A_3, A_4, A_5\} = \{a, b, c, d, e, f, g\} \setminus \{b, c, d, e, g\} = \{a, f\}$.

$\zeta(A_2) = B_2 = \bigcup_{A_t \in \{A_2 \cup \mathcal{S}^-(A_2)\}}(A_t \setminus \bigcup \mathcal{S}^+(A_t)) = (A_2 \setminus \{\emptyset\}) \cup (A_1 \setminus \bigcup\{A_2, A_3, A_4, A_5\}) = \{b, g\} \cup \{a, f\} = \{a, f, b, g\}$.

$\zeta(A_3) = B_3 = \bigcup_{A_t \in \{A_3 \cup \mathcal{S}^-(A_3)\}}(A_t \setminus \bigcup \mathcal{S}^+(A_t)) = (A_3 \setminus \{A_4, A_5\}) \cup (A_1 \setminus \bigcup\{A_2, A_3, A_4, A_5\}) = \{c\} \cup \{a, f\} = \{c, a, f\}$.

$\zeta(A_4) = B_4 = \bigcup_{A_t \in \{A_4 \cup \mathcal{S}^-(A_4)\}}(A_t \setminus \bigcup \mathcal{S}^+(A_t)) = (A_4 \setminus \{\emptyset\}) \cup (A_3 \setminus \{A_4, A_5\}) \cup (A_1 \setminus \bigcup\{A_2, A_3, A_4, A_5\}) = \{d\} \cup \{c\} \cup \{a, f\} = \{d, c, a, f\}$.

$\zeta(A_5) = B_5 = \bigcup_{A_t \in \{A_5 \cup \mathcal{S}^-(A_5)\}}(A_t \setminus \bigcup \mathcal{S}^+(A_t)) = (A_5 \setminus \{\emptyset\}) \cup (A_3 \setminus \{A_4, A_5\}) \cup (A_1 \setminus \bigcup\{A_2, A_3, A_4, A_5\}) = \{e\} \cup \{c\} \cup \{a, f\} = \{e, c, a, f\}$.

In Figure 6.7 we can see a graphical representation of the NS-F mapped into a INS-F throughout the ζ *function.*

Now, let us see how the ξ function allows us to map an INS-F into a NS-F.

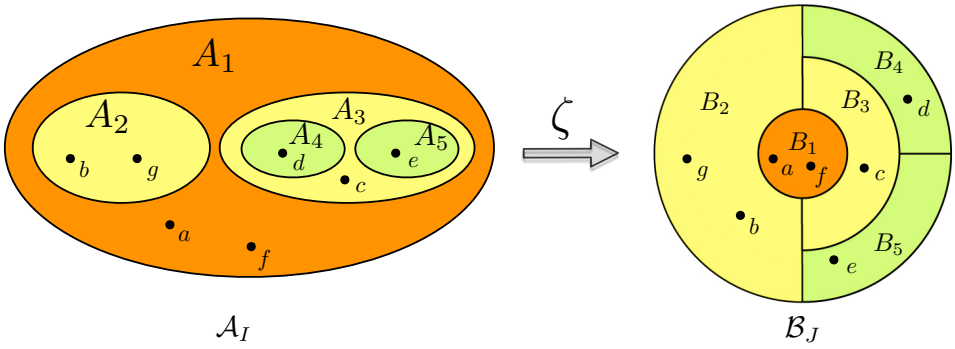


Figure 6.7: From the NS-F to the INS-f through the ζ function.

Theorem 6.7. *Let $\mathcal{A}_I$ be a INS-F then $\xi(\mathcal{A}_I) = \mathcal{B}_J$ is a NS-F.*

Proof:
To show that $\xi(\mathcal{A}_I) = \mathcal{B}_J$ is a NS-F, we have to prove the three conditions stated in Definition 6.1:

Condition (6.1): $\emptyset \notin \mathcal{B}_J$. *We know that* $\emptyset \notin \mathcal{A}_I \Rightarrow A_k \neq \emptyset, \forall A_k \in \mathcal{A}_I$ *and that* $\bigcup \mathcal{S}^+_{\mathcal{A}}(A_k) \subseteq (A_k \cup \bigcup \mathcal{S}^-(A_k))$ *by Definition 4.12. Thus* $B_k \neq \emptyset, \forall B_k \in \mathcal{B}_J$.

Condition (6.2): $B \notin \mathcal{B}_J$. *We know that* $\exists\, A_k \in \mathcal{A}_I \mid \forall A_j \in \mathcal{A}_I, A_k \subseteq A_j$. $((\bigcup(A_k \cup \mathcal{S}^-(A_k)) = \bigcup \mathcal{A}_I) \wedge (\bigcup \mathcal{S}^+(A_k) = \emptyset)) \Rightarrow B_k = \bigcup(A_k \cup \bigcup \mathcal{S}^-(A_k)) \setminus \mathcal{S}^+(A_k) = \bigcup \mathcal{A}_I \Rightarrow \exists B_k \in \mathcal{B} \mid \forall B_j \in \mathcal{B}_J, B_j \subseteq B_k$.

Condition (6.3): $\forall B_i, B_j \in \mathcal{B}_J, B_i \cap B_j \neq \emptyset \Rightarrow B_i \subset B_j \vee B_j \subset B_i$.

We consider $A_i, A_j \in \mathcal{A}_I \mid A_i \subset A_j$.
$\xi(A_i) = B_i = \bigcup(A_i \cup \mathcal{S}^-(A_i)) \setminus \bigcup \mathcal{S}^+(A_i)$.
$\xi(A_j) = B_j = \bigcup(A_j \cup \mathcal{S}^-(A_j)) \setminus \bigcup \mathcal{S}^+(A_j)$.
$A_i \subset A_j \Rightarrow \bigcup \mathcal{S}^-(A_j) \subset \bigcup \mathcal{S}^-(A_i) \wedge \bigcup \mathcal{S}^+(A_i) \subset \bigcup \mathcal{S}^+(A_j) \Rightarrow \bigcup(A_j \cup \mathcal{S}^-(A_j)) \subset \bigcup(A_i \cup \mathcal{S}^-(A_i)) \Rightarrow \forall i, j \in I, B_j \subset B_i \mid A_i \subset A_j$. *Ab absurdo suppose that* $\exists B_i, B_j \in \mathcal{B}_J \mid (B_i \cap B_j \neq \emptyset) \wedge (B_i \not\subseteq B_j) \wedge (B_j \not\subseteq B_i) \Rightarrow \exists B_k \in \mathcal{B} \mid B_k \subset B_i \wedge B_k \subset B_j \Rightarrow \exists A_k \in \mathcal{A} \mid (A_i \subset A_k) \wedge (A_j \subset A_k) \wedge (A_i \not\subseteq A_j) \wedge (A_j \not\subseteq A_i)$ *but this means that* $\exists\, C_W \subseteq \mathcal{A}_I \mid C_W$ *is a topped INS-F and at the same time it is not a linearly ordered family* ($\forall C_t, C_z \in C_W$. $C_t \subset C_z \vee C_z \subset C_t$). □

Example 6.7. *Let $\mathcal{A}_I$ be a INS-F and let $\mathcal{A}_I = \{A_1, A_2, A_3, A_4, A_5\}$ where $A_1 = \{a, f\}$, $A_2 = \{a, b, f, g\}$, $A_3 = \{c, a, f\}$, $A_4 = \{d, c, a, f\}$ and $A_5 = \{e, c, a, f\}$. We can see a graphical representation of this INS-F in the left part of the Figure 6.8.*

If we consider the subfamily $C_K \subset \mathcal{A}_I$ where $C_K = \{A_1, A_3, A_4\}$, we can see that C_K is a INS-F topped by the set A_4 and it is a linearly ordered family; this can be verified for all the

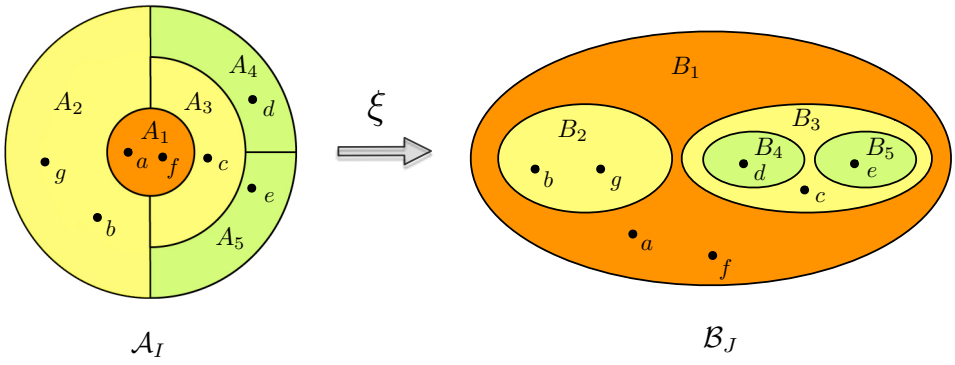


Figure 6.8: From the INS-F to the NS-F through the ξ function.

possible subfamilies of $\mathcal{A}_I$, thus $\xi(\mathcal{A}_I) = \mathcal{B}_J$ where $\mathcal{B}_J$ is a NS-F. So, if we apply the ξ function we obtain the following result:

$\xi(A_1) = B_1 = \bigcup(A_1 \cup \mathcal{S}_{\mathcal{A}^-}(A_1)) \setminus \bigcup \mathcal{S}^+(A_1) = \bigcup\{A_1, A_2, A_3, A_4, A_5\} \setminus \emptyset = \{a, b, c, d, e, f, g\}$.

$\xi(A_2) = B_2 = \bigcup(A_2 \cup \mathcal{S}^-_{\mathcal{A}}(A_2)) \setminus \bigcup \mathcal{S}^+(A_2) = \bigcup\{A_2\} \setminus A_1 = \{a, f, b, g\} \setminus \{a, f\} = \{b, g\}$.

$\xi(A_3) = B_3 = \bigcup(A_3 \cup \mathcal{S}^-_{\mathcal{A}}(A_3)) \setminus \bigcup \mathcal{S}^+(A_3) = \bigcup\{A_3, A_4, A_5\} \setminus A_1 = \{c, a, f, d, e\} \setminus \{a, f\} = \{c, d, e\}$.

$\xi(A_4) = B_4 = \bigcup(A_4 \cup \mathcal{S}^-_{\mathcal{A}}(A_4)) \setminus \bigcup \mathcal{S}^+(A_4) = A_4 \setminus \bigcup\{A_3, A_1\} = \{d, c, a, f\} \setminus \{c, a, f\} = \{d\}$.

$\xi(A_5) = B_5 = \bigcup(A_5 \cup \mathcal{S}^-_{\mathcal{A}}(A_5)) \setminus \bigcup(A_5) = A_5 \setminus \bigcup\{A_3, A_1\} = \{e, c, a, f\} \setminus \{c, a, f\} = \{e\}$.

As it is shown in the right part of Figure 6.8 $\mathcal{B}_J = \{B_1, B_2, B_3, B_4, B_5\}$ is a NS-F.

6.4 The Relationships Between the Set Data Models and the Tree Data Model

It is interesting to understand if and how we can map a tree into a family of sets defined by the set data models and vice versa. This feature is particularly important because it allows the interoperability between these data models; indeed, in a particular environment we can represent hierarchies by means of the set data models but this choice does not preclude possible relationships with other environments where the tree is adopted. Furthermore, we can use the set data models and the tree in the same context without any conflicts.

This section provides formal definitions of the mapping of a tree into a NS-F and vice versa as well as the mapping between a tree and a INS-F and vice versa. Furthermore, these mappings are fundamental to put in relation the operations performed in trees with their correspondents in the set data models. We show that none of the features of the tree is lost in the mapping to one of the set data models as well as that the tree expressive power is preserved.

6.4.1 The Nested Sets Model and the Tree Data Model

First-of-all let us see how we can map a tree into a NS-F. The following theorem formally defines and proves the intuitive mapping between a tree and a NS-F presented in the Section

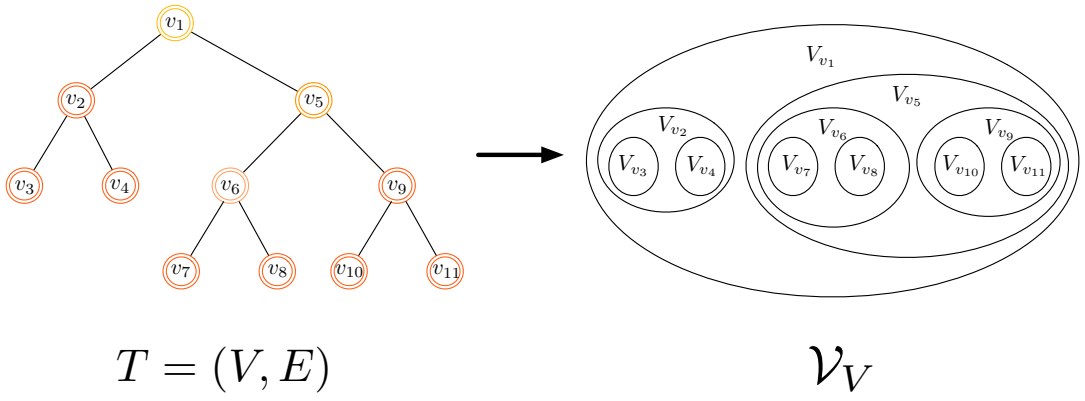


Figure 6.9: A tree $T = (V, E)$ and a NS-F $\mathcal{V}_V$ mapped from it.

3.2.1. It proves that a tree can be mapped into a NS-F by creating a set for each node and by defining the inclusion order between the newly created sets using the information brought by the edges connecting the nodes of the tree. For instance, let $T = (V, E)$ be a tree; if we consider an edge $e_{j,k} \in E$, then we have to create two sets A_j and A_k which correspond to the nodes $v_j, v_k \in V$ such that $A_k \subset A_j$; indeed, from $e_{j,k}$ we know that v_j is the parent of v_k and then the set A_j must be the superset of A_k.

Theorem 6.8. *Let $T = (V, E)$ be a tree and let $\mathcal{V}_V$ be a family where the set of nodes V is its index set of the family and $\forall v_i \in V$, $V_{v_i} = \Gamma^+(v_i)$. Then $\mathcal{V}_V$ is a Nested Set Family.*

Proof:
Let $v_r \in V$ be the root of the tree then $V_{v_r} = \Gamma^+(v_r) = V$ and thus $V \in \{V_{v_i}\}_{v_i \in V}$ (condition 6.1, Definition 6.1). By definition of descendant set of a node, $\forall v_i \in V$, $|V_{v_i}| = |\Gamma^+(v_i)| \geq 1$ and so $\emptyset \notin \mathcal{V}_V$ (condition 6.2, Definition 6.1).

Now, we prove condition 6.3 of Definition 6.1. Let $v_h, v_k \in V$, $h \neq k$ such that $V_{v_h} \cap V_{v_k} = \Gamma^+(v_h) \cap \Gamma^+(v_k) \neq \emptyset$, ab absurdo suppose that $\Gamma^+(v_h) \nsubseteq \Gamma^+(v_k) \ \wedge \ \Gamma^+(v_k) \nsubseteq \Gamma^+(v_k)$. This means that the descendants of v_h share at least a node with the descendants of v_k but they do not belong to the same subtree. This means that $\exists\, v_z \in V \mid d_V^-(v_z) = 2$ but then $T = (V, E)$ is not a tree. □

This theorem shows us that if we map a tree into a family of sets following the described rules, we obtain a NS-F.

Example 6.8. *Let $T = (V, E)$ be a tree where $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$ and $E = \{e_{1,2}, e_{1,5}, e_{2,3}, e_{2,4}, e_{5,6}, e_{5,9}, e_{6,7}, e_{6,8}, e_{9,10}, e_{9,11}\}$, thus $\Gamma^+(v_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_2) = \{v_2, v_3, v_4\}$, $\Gamma^+(v_3) = \{v_3\}$, $\Gamma^+(v_4) = \{v_4\}$, $\Gamma^+(v_5) = \{v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_6) = \{v_6, v_7, v_8\}$, $\Gamma^+(v_7) = \{v_7\}$, $\Gamma^+(v_8) = \{v_8\}$, $\Gamma^+(v_9) = \{v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_{10}) = \{v_{10}\}$, and $\Gamma^+(v_{11}) = \{v_{11}\}$.*

Let $\mathcal{V}_V$ be a family, where $V = \{V_{v_1}, V_{v_2}, V_{v_3}, V_{v_4}, V_{v_5}, V_{v_6}, V_{v_7}, V_{v_8}, V_{v_9}, V_{v_{10}}, V_{v_{11}}\}$, $V_{v_1} = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $V_{v_2} = \{v_2, v_3, v_4\}$, $V_{v_3} = \{v_3\}$, and $V_{v_4} = \{v_4\}$, $V_{v_5} = \{v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $V_{v_6} = \{v_6, v_7, v_8\}$, $V_{v_7} = \{v_7\}$, $V_{v_8} = \{v_8\}$, $V_{v_9} = \{v_9, v_{10}, v_{11}\}$, $V_{v_{10}} = \{v_{10}\}$, and $V_{v_{11}} = \{v_{11}\}$. Then, from Theorem 6.8 it follows that $\mathcal{V}_V$ is a NS-F.

The tree $T = (V, E)$ and the family $\mathcal{V}_V$ mapped from it are represented in Figure 6.9.

The following theorem shows that a NS-F can be mapped into a tree by creating a node from every set in the NS-F. Two sets A_j and A_k in the NS-F corresponds to two nodes v_j and v_k in the tree and the edge $e_{j,k}$ between them is created if and only if A_j is the direct superset of A_k.

Theorem 6.9. *Let $\mathcal{V}_V$ be a NS-F, V be a set of nodes and E be a set of edges where $\forall v_j \in V, \exists! V_{v_j} \in \mathcal{V}_V \wedge \forall e_{j,k} \in E, \exists!\ V_{v_j}, A_k \in \mathcal{V}_V \mid A_k \subset V_{v_j}$. Then $T = (V, E)$ is a tree.*

Proof:

We have to prove that $\exists! v_r \in V \mid |E^-(v_r)| = 0 \wedge \forall v_j \in V, j \neq r, |E^-(v_j)| = 1$. Ab absurdo suppose that $\exists v_r, v_k \in V \mid (|E^-(v_r)| = 0 \wedge |E^-(v_k)| = 0) \vee \exists v_j \in V \mid |E^-(v_j)| > 1$. If $\exists v_r, v_k \in V \mid |E^-(v_r)| = 0 \wedge |E^-(v_k)| = 0$ it means that both v_r and v_k have no ancestors $\Rightarrow\ \exists V_{v_r}, V_{v_k} \in \mathcal{V}_V \mid \mathcal{S}^-(V_{v_r}) = 0 \wedge \mathcal{S}^-(V_{v_k}) = 0$ but by the definition of NS-F we know that by definition there exists a set V_{v_w} such that every $V_{v_j} \in \mathcal{V}_V$ is a subset of the set V_{v_w} (i.e. V_{v_w} is the common superset of all the sets in a NS-F) and we also know that $V_{v_w} \in \mathcal{V}_V$ then $\nexists V_{v_j} \in \mathcal{V}_V \mid V_{v_j} \neq A \wedge \mathcal{S}^-(V_{v_j}) = 0$.

If $\exists v_j \in V \mid |E^-(v_j)| > 1$ this means that $\exists v_k, v_t \in V$ such that they are both parents of $v_j \Rightarrow\ \exists V_{v_k}, V_{v_t} \in \mathcal{V}_V \mid V_{v_j} \subset V_{v_k} \wedge V_{v_j} \subset V_{v_t} \Rightarrow (V_{v_t} \cap V_{v_k} = V_{v_j}) \wedge (V_{v_t} \nsubseteq V_{v_k} \vee V_{v_k} \nsubseteq V_{v_t}) \Rightarrow \mathcal{V}_V$ is not a NS-F. □

We have formally defined the relationships between a tree with the NS-M; we know that a tree can be mapped into a NS-F where every node of the tree is mapped into a set of the family. Now, we can compare the properties of the set data models with the correspondent ones in the tree. First-of-all we present a proposition describing the relationships between the NS-F properties and the tree ones.

Proposition 6.10. *Let $T = (V, E)$ be a tree, $\mathcal{V}_V$ be a NS-F, $V_{v_j}, V_{v_k}, V_{v_t} \in \mathcal{V}_V$, $j \neq k \neq t$ be three sets and $v_j, v_k, v_t \in V$, $j \neq k \neq t$ be the three correspondent nodes. If:*

$$V_{v_j} \cup V_{v_k} = V_{v_t} \Rightarrow \{v_j, v_k\} \in E^+(v_t) \tag{6.24}$$

$$V_{v_j} \cup V_{v_k} = V_{v_k} \Rightarrow v_k \in \Gamma^-(v_j) \tag{6.25}$$

$$V_{v_j} \cap V_{v_k} = V_{v_j} \Rightarrow v_j \in \Gamma^+(v_k) \tag{6.26}$$

$$V_{v_j} \cap V_{v_k} = \emptyset \Rightarrow v_j \notin \Gamma^+(v_k) \wedge v_j \notin \Gamma^-(v_k) \tag{6.27}$$

$$V_{v_j} \setminus V_{v_k} = V_{v_t} \Rightarrow \{v_k, v_t\} \in E^+(v_j) \tag{6.28}$$

$$V_{v_j} \setminus V_{v_k} = V_{v_j} \Rightarrow v_j \notin \Gamma^+(v_k) \wedge v_j \notin \Gamma^-(v_k) \tag{6.29}$$

Proof:

Property 6.24*: Ab absurdo suppose that* $V_{v_j} \cup V_{v_k} = V_{v_t} \Rightarrow \{v_j, v_k\} \notin E^+(v_t) \Rightarrow \nexists e_{t,j}, e_{t,k} \in E \Rightarrow V_{v_j} \nsubseteq V_{v_t} \wedge V_{v_k} \nsubseteq V_{v_t} \Rightarrow V_{v_j} \cup V_{v_k} = X \notin \mathcal{V}_V$.

Property 6.25*: Ab absurdo suppose that* $V_{v_j} \cup V_{v_k} = V_{v_k} \Rightarrow v_k \notin \Gamma^-(v_j) \Rightarrow V_{v_j} \nsubseteq V_{v_k} \Rightarrow V_{v_j} \cup V_{v_k} \neq V_{v_k}$.

Property 6.26*: Ab absurdo suppose that* $V_{v_j} \cap V_{v_k} = V_{v_j} \Rightarrow v_j \in \Gamma^+(v_k) \Rightarrow V_{v_j} \nsubseteq V_{v_k} \Rightarrow V_{v_j} \cap V_{v_k} \neq V_{v_j}$.

Property 6.27*: Ab absurdo suppose that* $V_{v_j} \cap V_{v_k} = \emptyset \Rightarrow v_j \in \Gamma^+(v_k) \vee v_j \in \Gamma^-(v_k)$. *If* $v_j \in \Gamma^+(v_k) \Rightarrow V_{v_j} \subset V_{v_k} \Rightarrow V_{v_j} \cap V_{v_k} \neq \emptyset$. *If* $v_j \in \Gamma^-(v_k) \Rightarrow V_{v_k} \subset V_{v_j} = V_{v_k} \neq \emptyset$.

Property 6.28*: Ab absurdo suppose that* $V_{v_j} \setminus V_{v_k} = V_{v_t} \Rightarrow \{v_k, v_t\} \notin E^+(v_j) \Rightarrow V_{v_k} \nsubseteq V_{v_j} \wedge V_{v_t} \nsubseteq V_{v_j} \Rightarrow V_{v_j} \setminus V_{v_k} = X \notin \mathcal{V}_V$

Property 6.29*: Ab absurdo suppose that* $V_{v_j} \setminus V_{v_k} = V_{v_j} \Rightarrow v_j \in \Gamma^+(v_k) \vee v_j \in \Gamma^-(v_k)$. *If* $v_j \in \Gamma^+(v_k) \Rightarrow V_{v_j} \subset V_{v_k} \Rightarrow V_{v_j} \setminus V_{v_k} = X \notin \mathcal{V}$. □

Property 6.24 states that if the union of two sets V_{v_j}, V_{v_k} in a NS-F $\mathcal{V}_V$ returns a third different set $V_{v_t} \in \mathcal{V}_V$ then, the correspondent nodes v_j and v_k in the tree must have the node v_t – which corresponds to the set V_{v_t} – as their parent node. This property is a consequence of Corollary 6.2; we can state the same for Property 6.28, indeed if $V_{v_j} \setminus V_{v_k} = V_{v_t} \in \mathcal{V}_V \Rightarrow V_{v_k} \cup V_{v_t} = V_{v_j}$ then v_j is the parent node of v_k and v_t.

Property 6.25 describes the particular case in which the union of two sets V_{v_j}, V_{v_k} in $\mathcal{V}_V$ returns V_{v_k}; it means that in the tree the node v_k is the parent of v_j. Property 6.26 states that if the intersection between two sets V_{v_j}, V_{v_k} in $\mathcal{V}_V$ is, for instance, V_{v_j} itself, it means that the correspondent node $v_j \in V$ is a descendant of v_k. Property 6.27 states that if the intersection between V_{v_j}, V_{v_k} is empty then the correspondent nodes v_j and v_k belong to two different branches of the tree; *we can establish if two nodes are in an ancestor-descendant relationship just by checking if the intersection of their correspondent sets in the NS-M is empty*. Lastly, Property 6.29 produces the same result of Property 6.27; indeed, $V_{v_j} \setminus V_{v_k} = V_{v_j} \Rightarrow V_{v_j} \cap V_{v_k} = \emptyset$.

Now, we take into account how some widely-known tree properties are mapped in the NS-M.

Proposition 6.11. *Let* $T = (V, E)$ *be a tree,* $v_i \in V$ *be a node, and* $\mathcal{V}_V$ *be a NS-F, then the following properties hold:*

$$\exists! v_r \in V \mid |E^-(v_r)| = 0 \Rightarrow \exists! V_{v_r} \in V \mid \mathcal{S}^-(V_{v_r}) = 0. \tag{6.30}$$

$$\forall v_j \in E^+(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{D}^+(V_{v_i}). \tag{6.31}$$

$$\forall v_j \in E^-(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{D}^-(V_{v_i}). \tag{6.32}$$

$$\forall v_j \in \Gamma^+(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{S}^+(V_{v_i}) \cup V_{v_i}. \tag{6.33}$$

$$\forall v_j \in \Gamma^-(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{S}^-(V_{v_i}) \cup V_{v_i}. \tag{6.34}$$

Proof:

Property 6.30. *By condition 6.1 of Definition 6.1,* $\exists! V_{v_j} \in \mathcal{V}_V \mid \forall V_{v_k} \in \mathcal{V}_V, k \neq j, V_{v_k} \subset V_{v_j} \Rightarrow \mathcal{S}^-(V_{v_j}) = 0$.

Property 6.31. *By Theorem 6.8 we know that* $\forall v_i \in V, \exists V_{v_i} \in \mathcal{V}_V = \Gamma^+(V_{v_i})$ *and that* $E^+(v_i) \subseteq \Gamma^+(v_i)$. *Then if* $\exists v_j \in E^+(v_i) \Rightarrow v_j \in \Gamma^+(v_i) \Rightarrow V_{v_j} \in \mathcal{V}_V$. *This means that* $\nexists v_k \in V, k \neq i \mid v_k \in E^+(v_j) \Rightarrow V_{v_j} \in \mathcal{D}^+(V_j)$.

Property 6.32. *Ab absurdo suppose that* $\exists v_j \in E^-(v_i) \mid V_{v_j} \notin \mathcal{D}^-(V_{v_i})$; *this means that* $\exists e_{j,i} \in E \mid \forall V_k \in V, k \neq j, \nexists e_{k,i} \in E$. *Furthermore, we know that* $v_j \in \Gamma^-(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{V}_V$ *if* $\nexists V_{v_j} \in \mathcal{D}^-(V_{v_i}) \Rightarrow \exists V_{v_k} \in \mathcal{V}_V \mid V_{v_i} \subset V_{v_k} \subset V_{v_j} \Rightarrow \exists e_{k,i} \in E \Rightarrow v_j \notin E^-(v_i)$.

Property 6.33. *Ab absurdo suppose that* $\exists v_j \in \Gamma^+(v_i) \mid \exists V_{v_j} \notin \mathcal{S}^+(V_{v_i}) \cup V_{v_i} \Rightarrow V_{v_j} \nsubseteq V_{v_i}$. *If* $V_{v_j} \nsubseteq V_{v_i}$ *it does not exist a path* $P = (V_P, E_P)$ *from* v_i *to* v_j *in* T, *so* $v_i P v_j = \emptyset \Rightarrow v_j \notin \Gamma^+(v_i)$.

Property 6.34. *Ab absurdo suppose that* $\exists v_j \in \Gamma^-(v_i) \mid \exists V_{v_j} \notin \mathcal{S}^-(V_{v_i}) \cup V_{v_i} \Rightarrow V_{v_i} \nsubseteq V_{v_j}$. *If* $V_{v_i} \nsubseteq V_{v_j}$ *it does not exist a path* $P = (V_P, E_P)$ *from* v_j *to* v_i *in* T, *so* $v_j P v_i = \emptyset \Rightarrow v_j \notin \Gamma^-(v_i)$. □

This proposition proves several properties of a NS-F relating them to the properties of a tree. Property 6.30 shows that there exists a unique set that has no supersets and that if the NS-F is mapped into a tree, this set corresponds to the root. Property 6.31 shows that determining the collection of direct subsets of a set in a NS-F corresponds to the operation of determining all the children of the corresponding node in a tree. We proved (Property 6.32) that the collection of direct supersets of a set corresponds to the parent of the corresponding node in a tree. In the same way (Property 6.32 and Property 6.33) we showed that the union of the collection of proper subsets (supersets) of a set with the set itself corresponds to the set of all the descendants (ancestors) of the corresponding node.

An important operation performed in the tree data structure is to determine the *lowest common ancestor* (lca) of two nodes – please refer to Section 4.2.4. In the following we define the concept of lowest common ancestor in the NS-M.

Definition 6.5.

Let $\mathcal{A}_I$ *be a NS-F,* $A_j, A_k \in \mathcal{A}_I$ *be two sets. Then,* $A_t \in \mathcal{A}_I$ *is defined to be the* ***lowest common ancestor*** *of* A_j *and* A_k *in* $\mathcal{A}_I$, *say* $A_t = \mathsf{lca}_{\mathcal{A}}(A_j, A_k)$, *if:*

$$A_t \in (\mathcal{S}^-(A_j) \cap \mathcal{S}^-(A_k)), \text{ and} \tag{6.35}$$

$$\nexists A_w \in ((\mathcal{S}^-(A_j) \cup A_j) \cap (\mathcal{S}^-(A_k) \cup A_k)) \mid |\mathcal{S}^+(A_w)| < |\mathcal{S}^+(A_t)|. \tag{6.36}$$

The lca in the NS-M is defined symmetrically to the lca in the tree data model. Indeed, in a tree the lca between two nodes is determined by the common ancestor between them with the maximum depth in the tree; the concept is specular in the NS-M where the lca between two sets, say A_j and A_k, is its common superset (Condition 6.35), say A_t, that has no subset which is also a superset of both A_j and A_k (Condition 6.36). We can also say that A_t is the $\mathsf{lca}_{\mathcal{A}}(A_j, A_k)$ if A_t is the common superset with the smaller cardinality among all their common supersets.

The following preposition shows how the concept of lca between two sets in a NS-M is related to the lca between two nodes in a tree.

Proposition 6.12. *Let $T = (V, E)$ be a tree, $v_j, v_k, v_t \in V$ be three nodes, $\mathcal{V}_V$ be a NS-F, and $V_{v_j}, V_{v_k}, V_{v_t} \in \mathcal{V}_V$ be three sets. Then,*

$$v_t = \mathsf{lca}_V(v_j, v_k) \Leftrightarrow V_{v_t} = \mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k})$$

Proof:
Let $C = (\mathcal{S}^-(V_{v_j}) \cup V_{v_j}) \cap (\mathcal{S}^-(V_{v_k}) \cup V_{v_k})$ be a collection of sets.

Let us prove ($\Rightarrow$). Ab absurdo suppose that $v_t = \mathsf{lca}_V(v_j, v_k)$ but $V_{v_t} \neq \mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k})$. This means that $\exists V_{v_w} \in C \mid |\mathcal{S}^+(V_{v_w})| < |\mathcal{S}^+(V_{v_t})| \Rightarrow \exists v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k) \mid |\Gamma^+(v_w)| < |\Gamma^+(v_k)| \Rightarrow v_t \neq \mathsf{lca}_V(v_j, v_k)$.

Let us prove ($\Leftarrow$). Ab absurdo suppose that $V_{v_t} = \mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k})$ but $v_t \neq \mathsf{lca}_V(v_j, v_k)$. This means that $\forall v_l \in V, \exists v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k), w \neq t \mid |\Gamma^+(v_w)| < |\Gamma^+(v_l)| \Rightarrow \forall V_{v_l} \in \mathcal{V}_V, \exists V_{v_w} \in C, v_w \neq v_l \mid |\mathcal{S}^+(V_{v_w})| < |\mathcal{S}^+(V_{v_l})| \Rightarrow V_{v_t} \neq \mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k})$. □

This proposition shows that if we map a tree into a correspondent NS-F also the nodes of the tree are mapped into sets in the family and thus the lca between two nodes is mapped into the lca between the correspondent sets. Furthermore, we can see that the lca between two sets in the NS-M can be determined *by taking the sets with minimum cardinality in the intersection between their collections of supersets*.

Example 6.9. *Let $T = (V, E)$ be the tree represented in Figure 6.10, and let $\mathcal{V}_V$ the NS-F mapped from T. If we consider the nodes v_7 and v_{11}, the $\mathsf{lca}_V(v_7, v_{11}) = v_5$ because the path $v_7 P v_1$ intersected with the path $v_{11} P v_1$ returns two nodes: v_1 and v_5; v_1 is the root and by definition its depth is 0, instead v_5 has depth 1 thus, it is the lowest common ancestor of v_7 and v_{11}.*

We consider the sets V_{v_7} and $V_{v_{11}}$ in $\mathcal{V}_V$; then, $\mathcal{S}^-(V_{v_7}) \cup V_{v_7} = \{V_{v_6}, V_{v_5}, V_{v_1}, V_{v_7}\}$ and $\mathcal{S}^-(V_{v_{11}}) \cup V_{v_{11}} = \{V_{v_6}, V_{v_5}, V_{v_1}, V_{v_{11}}\}$ thus, their intersection is the collection $C = \{V_{v_5}, V_{v_1}\}$.

Then, we can see that $\mathcal{S}^+(V_{v_5}) = \{V_{v_6}, V_{v_7}, V_{v_8}\}$ and so: $|\mathcal{S}^+(V_{v_5})| = 3$; instead $\mathcal{S}^+(V_{v_1}) = \{V_{v_5}, V_{v_6}, V_{v_7}, V_{v_8}\}$ and so: $|\mathcal{S}^+(V_{v_1})| = 4$. $|\mathcal{S}^+(V_{v_5})| < |\mathcal{S}^+(V_{v_1})|$, then $\mathsf{lca}_{\mathcal{V}}(V_{v_7}, V_{v_{11}}) = V_{v_5}$.

Figure 6.10 shows all the steps to determine the lca in T and in $\mathcal{V}_V$.

6.4.2 The Inverse Nested Sets Model and the Tree Data Model

Now, we can define the correspondent theorems for the INS-M which show how a tree can be mapped into a INS-F and vice versa. The following theorem formalizes the intuitive explanation about the mapping of a tree into a INS-F that we have given in Section 3.2.1. Basically, every couple of nodes v_j and v_k is mapped into a couple of sets A_j and A_k. If there exists an edge between v_j and v_k, say $e_{j,k}$ then the the set A_j created from v_j is defined as a subset of the

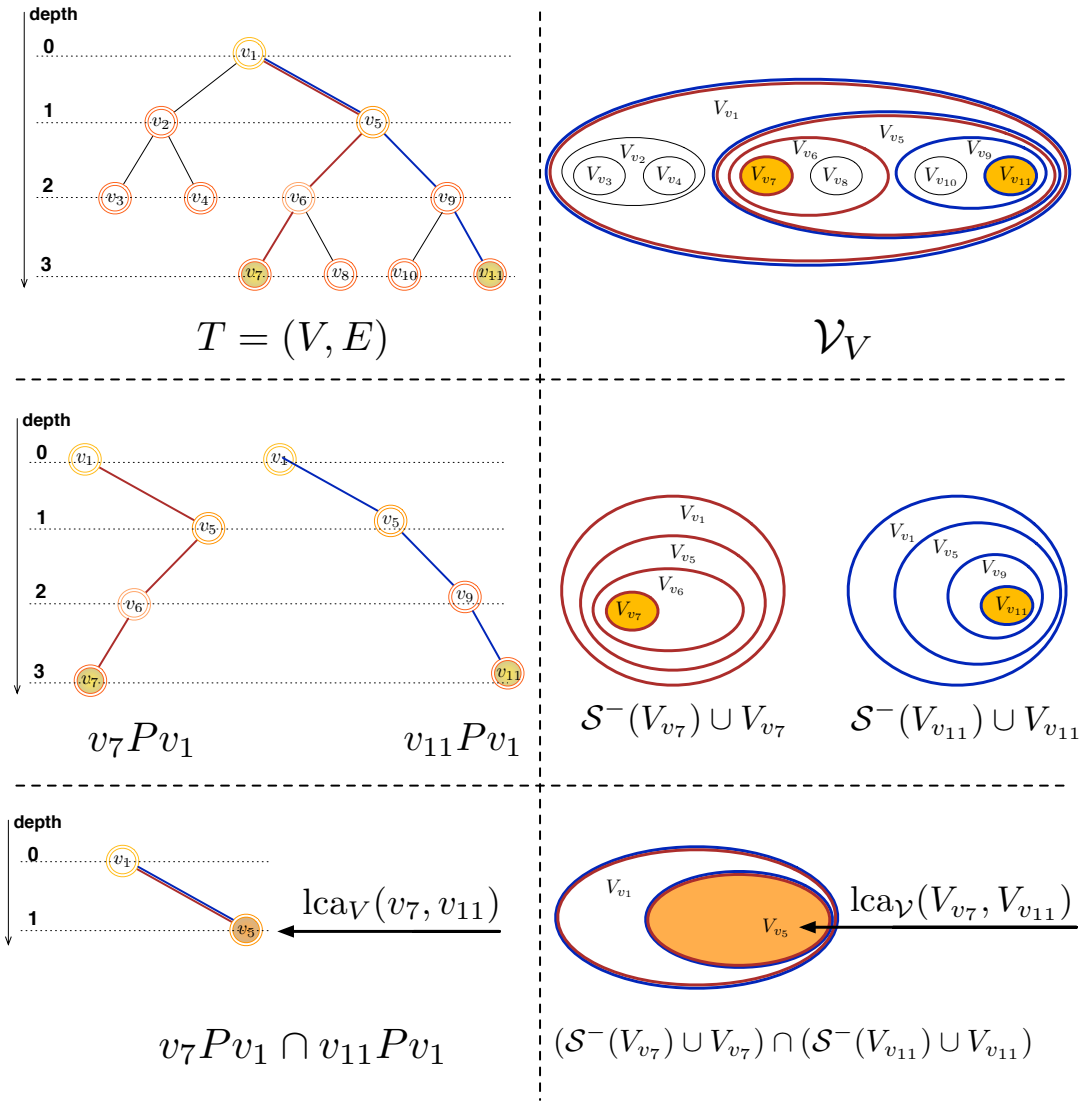


Figure 6.10: The lowest common ancestor in the tree $T = (V, E)$ and in the NS-F $\mathcal{V}_V$ mapped from it.

set A_k created from v_k. The mapping between a tree and an INS-F reverses the idea described for the mapping of a tree into a NS-F; if a node is parent of another node in a tree, this is mapped into a set which is a subset of the set created from its child node.

Theorem 6.13. *Let $T = (V, E)$ be a tree and let $\mathcal{V}_V$ be a family where the set of nodes V is its index set of the family and $\forall v_i \in V$, $V_{v_i} = \Gamma^-(v_i)$. Then $\mathcal{V}_V$ is an Inverse Nested Set family.*

Proof:

By definition of the set of the ancestors of a node, $\forall v_i \in V$, $|V_{v_i}| = |\Gamma^-(v_i)| \geq 1$ and so $\emptyset \notin \mathcal{V}_V$ (Condition 6.4 of Definition 6.2).

Let $\mathcal{V}_W$ be a subfamily of $\mathcal{V}_V \Rightarrow W \subseteq V$. We prove condition 6.5 by induction on the cardinality of W. $|W| = 1$ is the base case and it means that every subfamily $\mathcal{V}_W \subseteq \mathcal{V}_V$ is

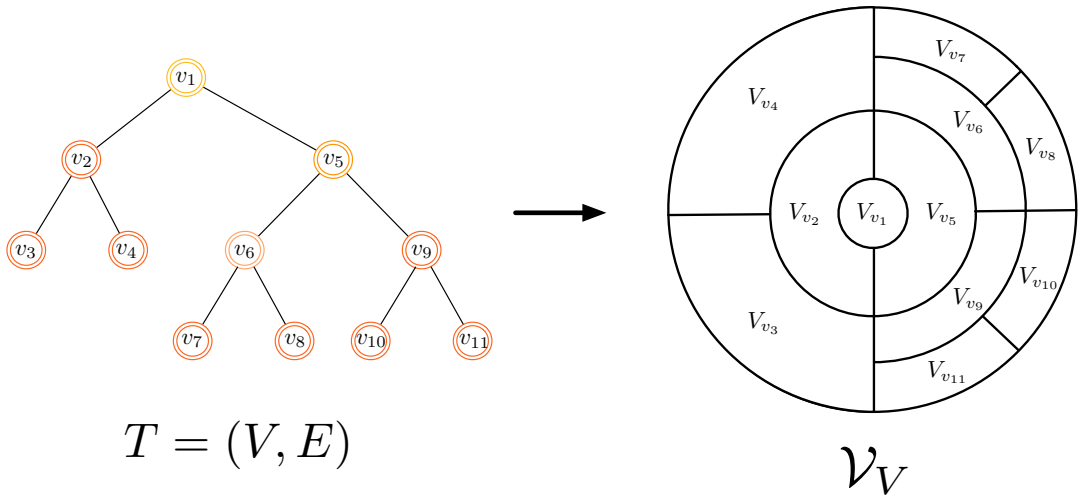


Figure 6.11: A tree $T = (V, E)$ and a INS-F $\mathcal{V}_V$ mapped from it.

composed only by one set V_{v_1} whose intersection is the set itself and belongs to the family $\mathcal{V}_V$ by definition.

For $|W| = n - 1$ we assume that $\exists\, v_{n-1} \in V \mid \bigcap_{v_j \in W} V_{v_j} = V_{v_{n-1}} \in \mathcal{V}_V$; equivalently we can say that $\exists\, v_{n-1} \in V \mid \bigcap_{v_j \in W} \Gamma^-(v_j) = \Gamma^-(v_{n-1})$, thus, $\Gamma^-(v_{n-1})$ is a set of nodes that is composed of common ancestors of the $n-1$ considered nodes.

For $|W| = n$, we have to show that $\exists\, v_t \in V \mid \forall\, v_n \in W,\ V_{v_{n-1}} \cap V_{v_n} = V_{v_t} \in \mathcal{V}_V$. This is equivalent to show that $\exists\, v_t \in V \mid \forall\, v_n \in W,\ \Gamma^-(v_{n-1}) \cap \Gamma^-(v_n) = \Gamma^-(v_t)$.

Ab absurdo suppose that $\exists\, v_n \in W \mid \forall\, v_t \in V,\ \Gamma^-(v_{n-1}) \cap \Gamma^-(v_n) \neq \Gamma^-(v_t)$. This would mean that v_n has no ancestors in W and, consequently, in V; at the same time, this would mean that v_n is an ancestor of no node in W and, consequently, in V. But this means that V is the set of nodes of a forest and not of a tree.

Now, we have to prove condition 6.6. Let $\mathcal{V}_W$ be a subfamily of $\mathcal{V}_V$. Ab absurdo suppose that $\exists V_{v_k} \in \mathcal{V}_W \mid \forall V_{v_h} \in \mathcal{V}_W, V_{v_h} \subseteq V_{v_k} \Rightarrow \exists V_{v_h}, V_{v_g} \in \mathcal{V}_W \mid V_{v_h} \nsubseteq V_{v_g} \wedge V_{v_g} \nsubseteq V_{V_h}$. This means that $\mathcal{V}_W$ is a topped but not linearly ordered family.

This means we can find $V_{v_g}, V_{v_h}, V_{v_k} \in \mathcal{V}_W \mid ((V_{v_h} \cap V_{v_k} \neq \emptyset) \wedge (V_{v_h} \cup V_{v_k} \subset V_{v_g}) \wedge (V_{v_h} \nsubseteq V_{v_k}) \wedge (V_{v_k} \nsubseteq V_{v_h})) \Rightarrow \exists v_h, v_k, v_g \in V \mid ((\Gamma^-(v_h) \cap \Gamma^-(v_k) \neq \emptyset) \wedge (\Gamma^-(v_h) \cup \Gamma^-(v_k \subseteq \Gamma^-(v_g)) \wedge (\Gamma^-(v_h) \nsubseteq \Gamma^-(v_k)) \wedge (\Gamma^-(v_k) \subseteq \Gamma^-(v_h)))$. This means that there are two paths from the root of T to v_g, one thought v_h and a distinct one thought v_k, thus $|E^-(v_g)| = 2$ and so T is not a tree. □

Example 6.10. *Let $T = (V, E)$ be a tree where $V = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$ and $E = \{e_{1,2}, e_{1,5}, e_{2,3}, e_{2,4}, e_{5,6}, e_{5,9}, e_{6,7}, e_{6,8}, e_{9,10}, e_{9,11}\}$, thus $\Gamma^+(v_1) = \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_2) = \{v_2, v_3, v_4\}$, $\Gamma^+(v_3) = \{v_3\}$, $\Gamma^+(v_4) = \{v_4\}$, $\Gamma^+(v_5) = \{v_5, v_6, v_7, v_8, v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_6) = \{v_6, v_7, v_8\}$, $\Gamma^+(v_7) = \{v_7\}$, $\Gamma^+(v_8) = \{v_8\}$, $\Gamma^+(v_9) = \{v_9, v_{10}, v_{11}\}$, $\Gamma^+(v_{10}) = \{v_{10}\}$, and $\Gamma^+(v_{11}) = \{v_{11}\}$.*

Let $\mathcal{V}_V$ be a family, where $V = \{V_{v_1}, V_{v_2}, V_{v_3}, V_{v_4}, V_{v_5}, V_{v_6}, V_{v_7}, V_{v_8}, V_{v_9}, V_{v_{10}}, V_{v_{11}}\}$, $V_{v_1} = \{v_1\}$, $V_{v_2} = \{v_1, v_2\}$, $V_{v_3} = \{v_1, v_2, v_3\}$, and $V_{v_4} = \{v_1, v_2, v_4\}$, $V_{v_5} = \{v_1, v_5\}$, $V_{v_6} = \{v_1, v_5, v_6\}$, $V_{v_7} = \{v_1, v_5, v_6, v_7\}$, $V_{v_8} = \{v_1, v_5, v_6, v_8\}$, $V_{v_9} = \{v_1, v_5, v_9\}$, $V_{v_{10}} = \{v_1, v_5, v_9, v_{10}\}$, and $V_{v_{11}} = \{v_1, v_5, v_9, v_{11}\}$. Then, from Theorem 6.8 it follows that $\mathcal{V}_V$ is a INS-F.

The tree $T = V, E$ and the family $\mathcal{V}_V$ mapped from it are represented in Figure 6.11.

Example 6.11. *Let $T = (V, E)$ be a tree where $V = \{v_0, v_1, v_2, v_3\}$ and $E = \{e_{0,1}, e_{0,2}, e_{2,3}\}$, thus $\Gamma^-(v_0) = \{v_0\}$, $\Gamma^-(v_1) = \{v_0, v_1\}$, $\Gamma^-(v_2) = \{v_0, v_2\}$ and $\Gamma^-(v_3) = \{v_0, v_2, v_3\}$. Let $\mathcal{V}_V$ be a family where $V_{v_0} = \{v_0\}$, $V_{v_1} = \{v_0, v_1\}$, $V_{v_2} = \{v_1, v_2\}$ and $V_{v_3} = \{v_0, v_2, v_3\}$.Then, from Theorem 6.13 it follows that $\mathcal{V}_V$ is a INS-F.*

Now we can see how an INS-M is mapped into a tree; the following theorem shows that if we map every couple of sets A_j and A_k in an INS-F into a couple of nodes v_j and v_k in a set of nodes V such that there exists an edge $e_{j,k}$ in a set of edges E if and only if A_j is a direct subset of A_k then the graph defined by the nodes in V connected by the edges in E is a tree.

Theorem 6.14. *Let $\mathcal{V}_V$ be a INS-F, V be a set of nodes and E be a set of edges where $\forall v_j \in V, \exists! V_{v_j} \in \mathcal{V}_V \wedge \forall e_{j,k} \in E, \exists! V_{v_j}, V_{v_k} \in \mathcal{V}_V \mid V_{v_j} \subset V_{v_k}$. Then $T = (V, E)$ is a tree.*

Proof:

We have to prove that $(\exists!\ v_r \in V \mid |E^-(v_r)| = 0) \wedge (\forall v_j \in V, j \neq r, |E^-(v_j)| = 1)$. Ab absurdo suppose that $\exists v_r, v_k \in V \mid (|E_V^-(v_r)| = 0 \wedge |E_V^-(v_k) = 0)| \vee \exists v_j \in V \mid |E_V^-(v_j)| > 1$.

If $\exists v_r, v_k \in V \mid |E^-(v_r)| = 0 \wedge |E^-(v_k)| = 0 \Rightarrow \exists V_{v_j}, V_{v_k} \in \mathcal{V}_V \mid V_{v_j} \cap V_{v_k} = \emptyset$ but $\emptyset \notin \mathcal{V}_V$ then $\mathcal{V}_V$ is not an INS-F.

If $\exists v_j \in V \mid |E_V^-(v_j)| > 1 \Rightarrow \exists V_{v_j}, V_{v_k}, V_{v_t} \in \mathcal{V}_V \mid (V_{v_k} \subset V_{v_j}) \wedge (V_{v_t} \subset V_{v_j}) \wedge (V_{v_k} \cap V_{v_t} = \emptyset) \Rightarrow \exists \mathcal{V}_W \subseteq \mathcal{V}_V$ where $\mathcal{V}_W = \{V_{v_j}, V_{v_k}, V_{v_t}\}$ and $\mathcal{V}_W$ is topped by V_{v_j} but it is not a linearly-ordered family – i.e. $V_{v_k} \cap V_{v_t} = \emptyset$ – thus, $\mathcal{V}_V$ is not a INS-F. □

The following proposition reflects for an INS-F the Proposition 6.10 we presented for a NS-F. From this proposition we see which set-theoretic operations defined in the INS-M find a correspondent property in the tree.

Proposition 6.15. *Let $T = (V, E)$ be a tree, $\mathcal{V}_V$ be a INS-F, $V_{v_j}, V_{v_k}, V_{v_t} \in \mathcal{V}_V, j \neq k \neq t$ be three sets in $\mathcal{V}_V$ and $v_j, v_k, v_t \in V, j \neq k \neq t$ be the three correspondent nodes in T. Then:*

$$V_{v_j} \cup V_{v_k} = V_{v_k} \Rightarrow v_k \in \Gamma^+(v_j) \tag{6.37}$$

$$V_{v_j} \cap V_{v_k} = V_{v_j} \Rightarrow v_j \in \Gamma^-(v_k) \tag{6.38}$$

$$V_{v_j} \cap V_{v_k} = V_{v_t} \Rightarrow v_t \in \Gamma^-(v_k) \cap \Gamma^-(v_j) \tag{6.39}$$

Proof:

Property 6.37. *Ab absurdo suppose that* $V_{v_j} \cup V_{v_k} = V_{v_k} \Rightarrow v_k \notin \Gamma^+(v_j) \Rightarrow V_{v_j} \nsubseteq V_{v_k} \Rightarrow V_{v_j} \cup V_{v_k} \neq V_{v_k}$.

Property 6.38. *Ab absurdo suppose that* $V_{v_j} \cap V_{v_k} = V_{v_j} \Rightarrow v_j \notin \Gamma^-(v_k) \Rightarrow V_{v_j} \nsubseteq V_{v_k} \Rightarrow V_{v_j} \cap V_{v_k} \neq V_{v_j}$.

Property 6.39. *Ab absurdo suppose that* $V_{v_j} \cap V_{v_k} = V_{v_t} \Rightarrow v_t \notin \Gamma^-(v_k) \cap \Gamma^-(v_j) \Rightarrow V_{v_t} \nsubseteq V_{v_j} \wedge V_{v_t} \nsubseteq V_{v_k} \Rightarrow V_{v_t} \notin \mathcal{S}^-(V_{v_j}) \wedge V_{v_t} \notin \mathcal{S}^-(V_{v_k}) \Rightarrow V_{v_j} \cap V_{v_k} \neq V_{v_t} \vee V_{v_j} \cap V_{v_k} = V_{v_t} \notin \mathcal{V}_V$. □

Property 6.37 shows that if the union of two sets $V_{v_j}, V_{v_k} \in \mathcal{V}_V$ returns V_{v_j} it means that $v_j \in V$ is a descendant of $v_k \in V$; this property is a direct consequence of the definition of INS-F. Property 6.38 shows that if the intersection of two sets $V_{v_j}, V_{v_k} \in \mathcal{V}_V$ returns V_{v_j}, it means that $v_j \in V$ is an ancestor of $v_k \in V$. Property 6.39 points out an interesting result: if the intersection of two sets $V_{v_j}, V_{v_k} \in \mathcal{V}_V$ returns a third set $V_{v_t} \in \mathcal{V}_V$, then this set corresponds to a common ancestor v_t of the nodes v_j and v_k; we will prove in Proposition 6.17 that v_t is the lowest common ancestor.

The following proposition shows that the important properties of the trees have a correspondent property in INS-M.

Proposition 6.16. *Let* $T = (V, E)$ *a tree,* $v_i \in V$ *a node, and* $\mathcal{V}_V$ *be a INS-F, then:*

$$\exists!\ v_r \in V \mid |E_V^-(v_r)| = 0 \Rightarrow \exists! V_{v_r} \in V \mid |\mathcal{S}^+(V_{v_r})| = 0. \tag{6.40}$$

$$\forall v_j \in E^+(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{D}^-(V_{v_i}). \tag{6.41}$$

$$\forall v_j \in E^-(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{D}^+(V_{v_i}). \tag{6.42}$$

$$\forall v_j \in \Gamma^+(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{S}^-(V_{v_i}) \cup V_{v_i}. \tag{6.43}$$

$$\forall v_j \in \Gamma^-(v_i) \Rightarrow \exists V_{v_j} \in \mathcal{S}^+(V_{v_i}) \cup V_{v_i}. \tag{6.44}$$

Proof:

Property 6.40. *We prove this property by induction on the cardinality of* V. $|V| = 1$ *is the base case.* $\mathcal{V}_V$ *is composed by one set* V_{v_1} *then* $|\mathcal{S}^+(V_{v_1})| = 0$.

For $|V| = n$ *we assume that* $\exists! V_1 \in \{V_{v_i}\}_{v_i \in V} \mid \mathcal{S}^+(V_{v_1}) = 0$.

For $|V| = n + 1$ *we have to show that* $\exists! V_j \in \{V_{v_i}\}_{v_i \in V} \mid |\mathcal{S}^+(V_{v_j})| = 0$. *We know that* $|\mathcal{S}^+(V_{v_1})| = 0$. *Ab absurdo suppose that* $|\mathcal{S}^+(V_{v_{n+1}})| = 0 \Rightarrow V_{v_1} \cap V_{v_{n+1}} = \emptyset$ *but* $\emptyset \notin \{V_{v_i}\}_{v_i \in V}$ *then condition 6.5 of Definition 6.2 is violated* $\Rightarrow \{V_{v_i}\}_{v_i \in V}$ *is not an INS-F.*

Property 6.41. *Ab absurdo suppose that* $\exists v_j \in E^+(v_i) \Rightarrow V_{v_j} \notin \mathcal{D}^+(V_{v_i}) \Rightarrow (V_{v_j} \nsubseteq V_{v_i}) \vee (\exists V_{v_k} \in \mathcal{V}_V, k \neq j \mid V_{v_j} \subset V_{v_k} \subset V_{v_i}) \Rightarrow \nexists e_{i,j} \in E \Rightarrow v_j \in E^+(v_i)$.

Property 6.42. *Ab absurdo suppose that* $\exists v_j \in E^-(v_i) \Rightarrow V_{v_j} \notin \mathcal{D}^-(V_{v_i}) \Rightarrow (V_{v_i} \nsubseteq V_{v_j}) \vee (\exists V_{v_k} \in \mathcal{V}_V, k \neq j \mid V_{v_i} \subset V_{v_k} \subset V_{v_j}) \Rightarrow \nexists e_{j,i} \in E \Rightarrow v_j \in E^-(v_i)$.

Property 6.43. *Ab absurdo suppose that* $\exists v_j \in \Gamma^+(v_i) \Rightarrow V_{v_j} \notin \mathcal{S}^-(V_{v_i}) \cup V_{v_i} \Rightarrow V_{v_i} \nsubseteq V_{v_j}$. *If* $V_{v_i} \nsubseteq V_{v_j}$ *then it does not exist a path* $P = (V_P, E_P)$ *from* v_i *to* v_j *in* T, *so* $v_i P v_j = \emptyset \Rightarrow v_j \notin \Gamma^+(v_i)$.

Property 6.44. *Ab absurdo suppose that* $\exists v_j \in \Gamma^-(v_i) \Rightarrow V_{v_j} \notin \mathcal{S}^+_{\mathcal{V}}(V_{v_i}) \cup V_{v_i} \Rightarrow V_{v_j} \nsubseteq V_{v_i}$.

If $V_{v_j} \nsubseteq V_{v_i}$ then it does not exist a path $P = (V_P, E_P)$ from v_j to v_i in T, so $v_j P v_i = \emptyset \Rightarrow v_j \notin \Gamma^-(v_i)$. □

This proposition shows how the basic tree properties can be mapped in the INS-M as well as we did with the NS-M. Property 6.40 shows that there exists a unique set that has no subsets and that if the INS-F is mapped into a tree, this set corresponds to the root. Property 6.41 shows that determining the collection of direct supersets of a set in an INS-F corresponds to the operation of determining all the children of the corresponding node in a tree. We proved (Property 6.42) that the collection of direct subsets of a set corresponds to the parent of the corresponding node in a tree. In the same way (Property 6.42 and Property 6.43) we showed that the union of the collection of proper subsets (supersets) of a set with the set itself corresponds to the set of all the ancestors (descendants) of the corresponding node.

If we consider Proposition 6.11 we can see that there is a straightforward relationship between a tree and a NS-M; for instance, the children of a node in a tree correspond to the subsets of the set mapped from that node in the NS-M. The INS-M reverses this logic; indeed, the children of a node correspond to the supersets of the set mapped from that node in the INS-M. This characteristic can be exploited to provide alternative ways to perform operations on hierarchies coming up with different solutions. We can see a meaningful example considering how to determine the lowest common ancestor in the INS-M.

Definition 6.6.
*Let $\mathcal{A}_I$ be an INS-M, and $A_j, A_k, A_t \in \mathcal{A}_I$ be three sets. If $A_t = A_j \cap A_k$, then $\mathsf{lca}_{\mathcal{A}}(A_j, A_k) = A_t$ is defined to be the **lowest common ancestor** between A_j and A_k.*

The relationship between the lca in a tree and in an INS-F can be easily determined by exploiting the properties described in Proposition 6.16.

Proposition 6.17. *Let $T = (V, E)$ be a tree, $v_j, v_k, v_t \in V$ be three nodes, $\mathcal{V}_V$ be a INS-F and $V_{v_j}, V_{v_k}, V_{v_t} \in \mathcal{V}_V$ be three sets. Then:*

$$v_t = \mathsf{lca}_V(v_j, v_k) \Leftrightarrow V_{v_t} = \mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k}). \tag{6.45}$$

Proof:
From Definition 6.6 we know that: $\mathsf{lca}_{\mathcal{V}}(V_{v_j}, V_{v_k}) = A_j \cap A_k$.

Let us prove ($\Rightarrow$). Ab absurdo suppose that $v_t = \mathsf{lca}_V(v_j, v_k) \Rightarrow V_{v_t} \neq V_{v_j} \cap V_{v_k} \Rightarrow V_{v_t} \nsubseteq V_{v_j} \wedge V_{v_t} \nsubseteq V_{v_j} \Rightarrow V_{v_j} \notin \mathcal{S}^+_{\mathcal{V}}(V_{v_t}) \wedge V_{v_j} \notin \mathcal{S}^+_{\mathcal{V}}(V_{v_k}) \Rightarrow v_t \notin \Gamma^-(v_j) \wedge v_k \notin \Gamma^-(v_k)$.

Let us prove ($\Leftarrow$). Ab absurdo suppose that $V_{v_t} = V_{v_j} \cap V_{v_k} \Rightarrow v_t \neq \mathsf{lca}_V(v_j, v_k) \Rightarrow (v_t \notin \Gamma^-(v_j) \cap \Gamma^-(v_k)) \vee (\exists v_w \in V, w \neq t \mid (v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k)) \vee (v_w \in \Gamma^+(v_t)))$.

$v_t \notin \Gamma^-(v_j) \cap \Gamma^-(v_k) \Rightarrow v_t \notin \Gamma^-(v_j) \cap \Gamma^-(v_k) \Rightarrow V_{v_t} \notin (\mathcal{S}^+(V_{v_j}) \cup V_{v_j}) \cap (\mathcal{S}^+(V_{v_k} \cup V_{v_k})) \Rightarrow V_{v_t} \notin \mathcal{S}^-(V_{v_j}) \wedge V_{v_t} \notin \mathcal{S}^-(V_{v_k}) \Rightarrow V_{v_j} \cap V_{v_k} \neq V_{v_t}$.

$\exists v_w \in V, w \neq t \mid (v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k)) \vee (v_w \in \Gamma^+(v_t)) \Rightarrow v_t \in \Gamma^-(v_w) \Rightarrow V_{v_t} \subset V_{v_w} \Rightarrow V_{v_w} \in \mathcal{S}^-(V_{v_t}) \Rightarrow (V_{v_w} \subseteq V_{v_j} \cap V_{v_k}) \wedge (V_{v_t} \subseteq V_{v_j} \cap V_{v_k}) \Rightarrow V_{v_j} \cap V_{v_k} = V_{v_w}$. □

This proposition shows that if we map a tree into a correspondent INS-F also the nodes of the tree are mapped into sets in the family and thus the lca between two nodes is mapped into the lca between the correspondent sets. Furthermore, we can see that the lca between two sets in the INS-M can be determined *by the intersection of the considered sets*.

Example 6.12. *Let $T = (V, E)$ be a tree and let $\mathcal{V}_V$ a INS-F mapped from T. If we consider the nodes v_7 and v_{11} the $\mathsf{lca}_V(v_7, v_{11}) = v_5$ because the path $v_7 P v_1$ intersected with the path $v_{11} P v_1$ returns two nodes: v_1 and v_5; v_1 is the root and by definition its depth is 0, instead v_5 has depth 1 thus, it is the lowest common ancestor between v_7 and v_{11}; as we have seen in Figure 6.10 of Example 6.9.*

We consider the sets V_{v_7} and $V_{v_{11}}$ in $\mathcal{V}_V$ represented in Figure 6.11; we can see that V_{v_1} is a common subset of both V_{v_7} and $V_{v_{11}}$ as well as V_{v_5}. But $V_{v_1} \subset V_{v_5}$. Furthermore, $V_{v_7} \cap V_{v_{11}} = V_{v_5}$ which correspond to the node $v_5 \in V$ of the tree.

From this example we can see the correspondence between $\mathsf{lca}_V(v_7, v_{11})$ in T and $\mathsf{lca}_{\mathcal{V}}(V_{v_7}, V_{v_{11}})$ in $\mathcal{V}_V$.

6.5 Partially Ordered Sets and Families

We have presented two set data models which are defined by means of a bunch of constraints imposed on a family of sets. We have seen that by means of these set data models we can represent the tree by means of collections of nested sets. Our major focus has been the inclusion order between the sets; now, we have to analyze the possibility to represent the order between the subsets of a set and the relationships between the elements belonging to these sets.

The models we defined permit us to represent and determine the relationships between the elements and the set or sets containing them. For instance, we can say that n elements belong to a set A_2 which is subset of another set A_1 containing other m elements not contained in A_2. Thanks to the defined data models we know that n elements in A_2 elements are related to the m elements in A_1 by a hierarchical dependence; for instance, the n objects may be specializations of the m elements. On the other hand, when we consider the set A_1 we take into account $m + n$ elements, because A_1 contains also the n elements in A_2.

The models we presented so far permits us to represent the vertical dimension of a hierarchy by defining an inclusion order between the sets belonging to a family. By means of this logical construct we can determine the relationships between sets at different levels or perform operations with the sets; for instance, we can determine the lowest common ancestor (lca) of two or more sets. On the other hand, we have to take into consideration also the horizontal dimension of a hierarchy.

Indeed, a hierarchy always defines a hierarchical order between the elements it contains but

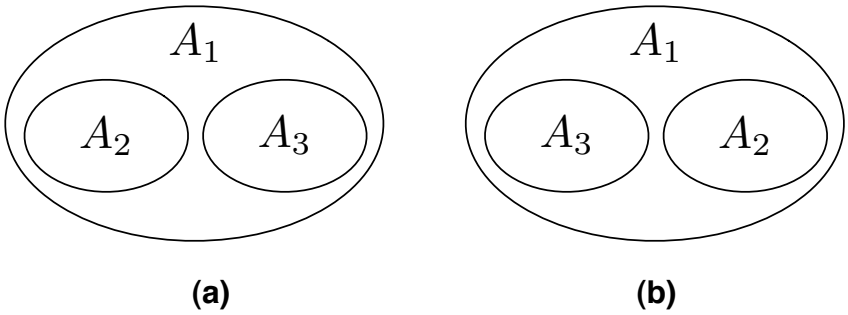


Figure 6.12: Two NS-families.

it may define also an order between the elements that lie at the same level of the hierarchy. For instance, a tree defines hierarchical relationships between its nodes – i.e. *rooted tree*, but often it defines also an order between the children of a node – i.e *ordered tree*.

If we need to represent the correspondent of an ordered tree in the NESTOR Model we have to take into account the order between the subsets of a set. In order to achieve this goal it is necessary to exploit the concept of *partially ordered set* (poset) – please refer to Section 4.1.5 for a description of this concept.

We use the notion of poset to define an order between the direct subsets of a set in a NS-F and an INS-F. In order to represent an ordered tree, we need to allow the possibility to define an order between the sets exploiting the concept of **partially ordered family of subsets** (see Section 4.1.5). Let $\mathcal{A}_I$ be a NS-F where I is its index set; I is a "non-ordered" set which means that we do not consider the order between the elements in it. As a consequence the sets in $\mathcal{A}_I$ are not ordered too; this means that the two NS-F represented in Figure 6.12 are the same family. Indeed, the set A_1 has two proper subsets A_2 and A_3, so $A_1 = \{A_2, A_3\} = \{A_3, A_2\}$.

If we consider an index poset $\langle I, < \rangle$ then $\mathcal{A}_{\langle I,< \rangle}$ results to be a **partially ordered NS-F** (poNSF-F) where the order between the common subsets of a set has to be taken into account. If the two NS-F in Figure 6.12 are indexed by $\langle I, < \rangle$ then they are different one to the other indeed, $A_1 = (A_2, A_3) \neq (A_3, A_2)$; this consideration comes straightforwardly from the definition of poset. In this case we say that A_1 is non-comparable with A_2 and A_3 ($A_1 \| A_2$ and $A_1 \| A_3$). The **partially ordered INS-F** (poINS-M) is defined exactly in the same way.

All the definitions and theorems proved for NS-F and INS-F can be extended by the use of partially ordered families without any changes in their formulation; we present the two following propositions straightforwardly derived from the definition of NS-F (Definition 6.1, Section 6.1) and INS-F (Definition 6.2, Section 6.1) and the definition of partially ordered family (Definition 4.19, Section 4.1.5).

The following two propositions show us that if the set of the nodes of a tree is defined by means of a poset that establishes a linear order between the nodes then this order is preserved also between the sets mapped from these nodes.

Proposition 6.18. *Let $T = (\langle V, < \rangle, E)$ be an ordered tree, $\{v_j, v_k, v_t\} \in V$ be three nodes, where*

$\{v_j, v_k\} \in E^+(v_t) \mid v_j < v_k$ *and* $\mathcal{V}_{\langle V,<\rangle}$ *be a poNS-F, then:*

$$\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle} \mid (\{V_{v_j}, V_{v_k}\} \in \mathcal{D}^+(V_{v_t})) \wedge (V_{v_j} < V_{v_k}).$$

Proof:
The fact that $\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle}$ *is a direct consequence of Theorem 6.8. Ab absurdo suppose that* $\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle} \mid (\{V_{v_j}, V_{v_k}\} \in \mathcal{D}^+(V_{v_t})) \wedge (V_{v_j} \not< V_{v_k}) \Rightarrow \exists\{v_j, v_k\} \in E^+(v_t) \mid (v_j \not< v_k)$*. Indeed, the order between sibling nodes in T is retained by the node set of T is the poset* $\langle V, <\rangle$.□

Proposition 6.19. *Let* $T(\langle V, <\rangle, E)$ *be an ordered tree,* $\{v_j, v_k, v_t\} \in V$ *be three nodes, where* $v_j, v_k \in E^+(v_t) \mid v_j < v_k$ *and* $\mathcal{V}_{\langle V,<\rangle}$ *be a poINS-F, then:*

$$\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle} \mid (\{V_{v_j}, V_{v_k}\} \in \mathcal{D}^-(V_{v_t})) \wedge (V_{v_j} < V_{v_k}).$$

Proof:
The fact that $\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle}$ *is a direct consequence of Theorem 6.13. Ab absurdo suppose that* $\exists\{V_{v_j}, V_{v_k}, V_{v_t}\} \in \mathcal{V}_{\langle V,<\rangle} \mid (\{V_{v_j}, V_{v_k}\} \in \mathcal{D}^-(V_{v_t})) \wedge (V_{v_j} \not< V_{v_k}) \Rightarrow \exists\{v_j, v_k\} \in E^+(v_t) \mid (v_j \not< v_k)$*. Indeed, the order between sibling nodes in T is retained by the node set of T is the poset* $\langle V, <\rangle$.□

The concept of poset can be exploited also to define the relationships between the items belonging to the sets in the nested sets models. We can extend the definitions of NS-F and INS-F by using posets instead of sets; in this case we talk of **Nested Posets Family** (NP-F) and **Inverse Nested Posets Family** (INP-F). From the definition of poset we can see that we do not need to re-define any concepts antecedently defined for NS-F and INS-F; *NP-F and INP-F add a new layer at the NESTOR Model allowing us to define a structure over the elements belonging to the sets*.

We can represent a NP-F by means of an Euler-Venn diagram in which the elements belonging to the sets are represented by means of an Hasse diagram; when no relation is drawn between the elements it means that we are talking about a NS-F. The same idea is applied to the DocBall representation for the INP-F and the INS-F. The following examples show the representations of a NP-F and of an INP-F and their correspondents NS-F and INS-F.

Example 6.13. *Let* $\mathcal{A}_I$ *be a NP-F and* $\langle A_1, <\rangle, \langle A_2, <\rangle, \langle A_3, <\rangle, \langle A_4, <\rangle \in \mathcal{A}_I$*.* $A_1 = \{a, b, c, d, e, f, g, h\}$ *where* $b < a, d < a, c < a, e < b, f < b, e < d, h < d, f < c, h < c, g < e, g < f$ *and* $g < h$*.* $A_2 = \{b, f, e, g\}$ *then* $A_2 \subset A_1$ *and it inherits the order relation from* A_1 *as well as* $A_3 = \{b, f\}$ *and* $A_4 = \{g, h\}$*. The four ordered sets are represented in Figure 6.13 throughout Hasse diagrams; in the part b of this Figure we can see the family NP-F and the correspondent NS-F that does not take into account the order between the items.*

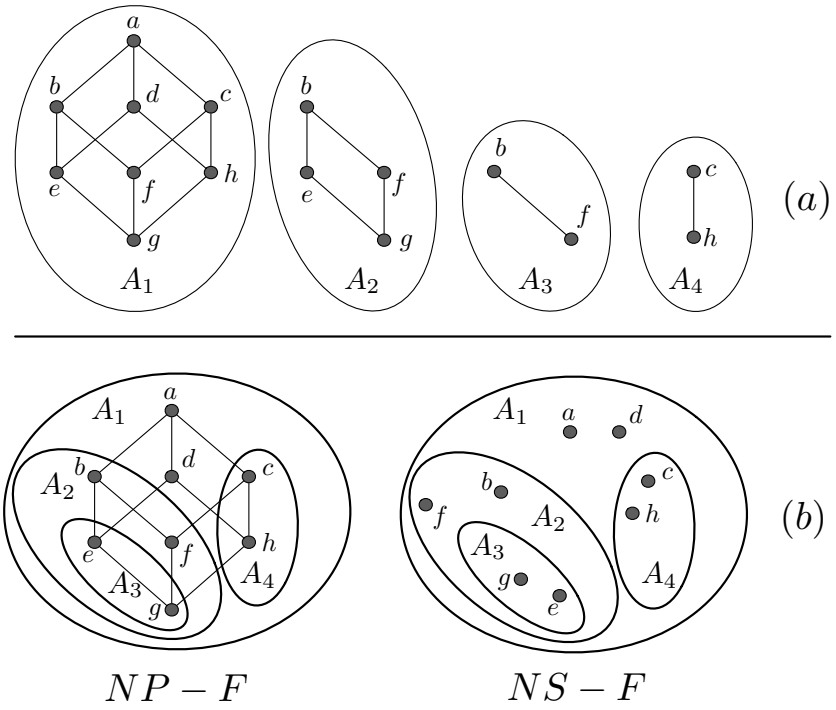


Figure 6.13: (a) Four partially ordered sets. (b) A NP-F composed of the four presented ordered sets and its correspondent NS-F

It is easy to tell from a Hasse diagram whether one element of an ordered set is less than another: $b < a$ if and only if there is a sequence of connected line segments moving upwards from b to a. Thus, for example, in the ordered set A_1 in Figure 6.13 $g < a$ and $c \parallel d$. The same representation can be adopted for the INP-F as it is showed in the following example.

Example 6.14. *Let $\mathcal{A}_I$ be a INP-F and $\langle A_1, <\rangle, \langle A_2, <\rangle, \langle A_3, <\rangle, \langle A_4, <\rangle \in \mathcal{A}_I$. $A_1 = \{a, d\}$ where $d < a$, $A_2 = \{a, b, d, f\}$ where $b < a, d < a, f < b$, $A_3 = \{a, b, d, f, e, g\}$ where $b < a, d < a, e < b, f < b, e < d, g < e, g < f$, and $A_4 = \{a, d, c, h\}$, where $d < a, c < a, h < d, h < c$.*

We can see that from the INP-F in the figure above the elements c and f and h and g are incomparable ($c \parallel f$ and $h \parallel g$). In the example 6.13 the set A_1 contains all the elements of the family – this is a direct consequence of condition 6.1 of Definition 6.1 ($A \in \mathcal{A}_I$) – thus we know all the order relationships between the elements from a A_1; instead, in the second example (Example 6.14) we have to consider the whole family of subsets to understand the order between all the elements in the family because, in an INS-F as well as in an INP-F, there is not a set containing all of them.

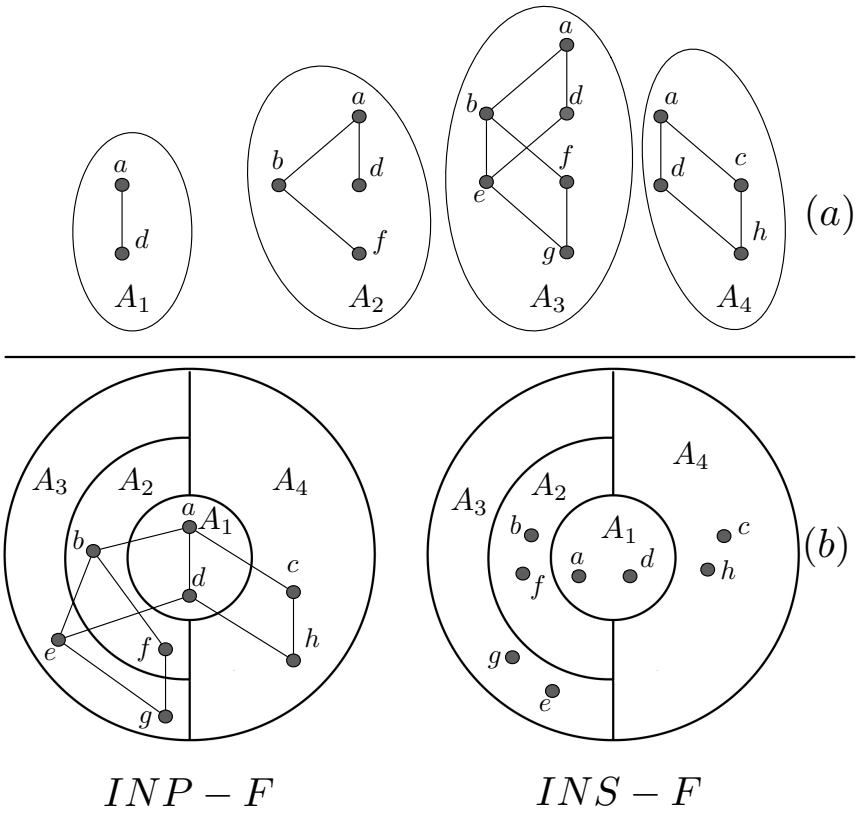


Figure 6.14: (a) Four partially ordered sets. (b) An INP-F composed by the four presented ordered sets and its correspondent INS-F

6.6 On the Metric Properties of the Set Data Models

In this section we define proper metrics for the set data models defined in the NESTOR Model. We have related the set data models to the tree defining the connections between these models. In this section we show how the metrics defined in the tree can be adapted and adopted by the set data models establishing a relation between them.

Firstly, we define the graphical distance for the NS-M and the INS-M which gives us the measure of how much dissimilar are two sets within a family of sets. Furthermore, we define the relationships between the graphical distance defined on the set data models and the correspondent distance in the tree.

Afterwards, we give broader definitions of distance which allow us to compare different families of sets. Two or more families of subsets can be "similar" from the "element" point-of-view and from the structural point-of-view. From the *element point-of-view* two families are identical if they contain the same elements. On the other hand, the structural part regards the inclusion order between the sets in each family; we can say that two families are identical from the *structural point-of-view* if there exists an order-isomorphism between them. This means that two families are identical if there exists an order-isomorphism between them and they contain the same elements.

We define a proper metric based on the Jaccard's distance to compare two families of sets on an element basis and a proper metric to compare families on a structural basis. In the end we present a linear combination of this two metrics that allows us to define a third proper metric which weights the element and the structural components: the NESTOR distance.

6.6.1 Graphical Distance

In graph theory and thus in the tree, one of the most used metric is the *graphical distance* (Definition 4.24 in Section 4.2.3). The graphical distance in a graph has been proved to be a proper metric; we define the graphical distance between two sets in the NS-M and in the INS-M. Afterwards, we prove that if we consider a NS-F $\mathcal{A}_I$ (or an INS-F) mapped into a tree $T = (V, E)$, the graphical distance between two sets $\{A_j, A_k\} \in \mathcal{A}_I$ is the same that the graphical distance between the nodes $\{v_j, v_k\} \in V$ mapped from A_j and A_k. In other words we show the equivalence between the graphical distance defined on trees and the graphical distance defined on the set data models.

Definition 6.7.
Let $\mathcal{A}_I$ be a NS-F, $\{A_j, A_k, A_t\} \in \mathcal{A}_I$ and, $A_t = \mathsf{lca}(A_j, A_k)$. Then we define the ***graphical distance*** *of A_j from A_k in $\mathcal{A}_I$ to be:*

$$d_G^{\mathcal{A}}(A_j, A_k) = |\mathcal{S}^-(A_j)| + |\mathcal{S}^-(A_k)| - 2|\mathcal{S}^-(A_t)| \tag{6.46}$$

Now, we can introduce a corollary of Proposition 6.11 that proves the correspondence between the tree graphical distance (d_G^T) and the NS-M graphical distance ($d_G^{\mathcal{A}}$).

Corollary 6.20. *Let $T = (V, E)$ be a tree, $\{v_j, v_k\} \in V$ be two nodes, $\mathcal{V}_V$ be a NS-F and $\{V_j, V_k\} \in \mathcal{V}_V$ be two sets. Let $d_G^T(v_j, v_k)$ be the graphical distance in the tree between v_j and v_k and $d_G^{\mathcal{V}}(V_{v_j}, V_{v_k})$ be the graphical distance in the NS-F between V_{v_j} and V_{v_k}. Then:*

$$d_G^T(v_j, v_k) = d_G^{\mathcal{V}}(V_{v_j}, V_{v_k}) \tag{6.47}$$

Proof:
Let us consider $v_t \in V = \mathsf{lca}(v_j, v_k)$; furthermore, let $l_{j,k}$ be the length of the path from v_j to v_k in T, $l_{j,t}$ be the length of the path from v_j to v_t in T, and $l_{t,k}$ be the length of the path from v_t to v_k in T.

Then $d_G^V(v_j, v_k) = l_{j,k} = l_{j,t} + l_{k,t} = |\Gamma^-(v_j)| - |\Gamma^-(v_t)| + |\Gamma^-(v_k)| - |\Gamma^-(v_t)| = |\Gamma^-(v_j)| + |\Gamma^-(v_k)| - 2|\Gamma^-(v_t)|$.

We know that in NS-M $\forall v_w \in V, \exists V_{v_w} \in \mathcal{V}_V \mid |\Gamma^-(v_t)| = |\mathcal{S}^-(A_w) \cup A_w|$. Then, $d_G^V(v_j, v_k) = |\Gamma^-(v_j)| + |\Gamma^-(v_k)| - 2|\Gamma^-(v_t)| = |\mathcal{S}^-(A_j) \cup A_j| + |\mathcal{S}^-(A_k) \cup A_k| - 2|\mathcal{S}^-(A_t) \cup A_t| = |\mathcal{S}^-(A_j)| + 1 + |\mathcal{S}^-(A_j)| + 1 - 2(|\mathcal{S}^-(A_t)| + 1) = |\mathcal{S}^-(A_j)| + |\mathcal{S}^-(A_j)| - 2|\mathcal{S}^-(A_t)| = d_G^{\mathcal{V}}(V_{v_j}, V_{v_k})$. □

Symmetrically, we present the definition of graphical distance between sets in the INS-M.

Definition 6.8.
Let $\mathcal{A}_I$ be a INS-F, $\{A_j, A_k, A_t\} \in \mathcal{A}_I$ and, $A_t = \mathsf{lca}(A_j, A_k)$. Then we define the ***graphical distance*** *of A_j from A_k in $\mathcal{A}_I$ to be:*

$$d_G^{\mathcal{A}}(A_j, A_k) = |\mathcal{S}^+(A_j)| + |\mathcal{S}^+(A_k)| - 2|\mathcal{S}^+(A_t)| \tag{6.48}$$

Now, we can introduce a corollary to Proposition 6.16 that proves the correspondence between the tree graphical distance and the graphical distance between two sets in the INS-M.

Corollary 6.21. *Let $T = (V, E)$ be a tree, $\{v_j, v_k\} \in V$ be two nodes, $\mathcal{V}_V$ be a INS-F and $\{V_j, V_k\} \in \mathcal{V}_V$ be two sets. Let $d_G^T(v_j, v_k)$ be the distance in the tree between v_j and v_k and $d_G^{\mathcal{V}}(V_{v_j}, V_{v_k})$ be the distance in the INS-F between V_{v_j} and V_{v_k}. Then:*

$$d_G^T(v_j, v_k) = d_G^{\mathcal{V}}(V_{v_j}, V_{v_k}) \tag{6.49}$$

Proof:
Let us consider $v_t \in V = \mathsf{lca}(v_j, v_k)$; furthermore, let $l_{j,k}$ be the length of the path from v_j to v_k in T, $l_{j,t}$ be the length of the path from v_j to v_t in T, and $l_{t,k}$ be the length of the path from v_t to v_k in T.

Then $d_G^V(v_j, v_k) = l_{j,k} = l_{j,t} + l_{k,t} = |\Gamma^-(v_j)| - |\Gamma^-(v_t)| + |\Gamma^-(v_k)| - |\Gamma^-(v_t)| = |\Gamma^-(v_j)| + |\Gamma^-(v_k)| - 2|\Gamma^-(v_t)|$.

We know that in the INS-M $\forall v_w \in V, \exists V_{v_w} \in \mathcal{V}_V \mid |\Gamma^-(v_t)| = |\mathcal{S}^+(A_w) \cup A_w|$. Then, $d_G^V(v_j, v_k) = |\Gamma^-(v_j)| + |\Gamma^-(v_k)| - 2|\Gamma^-(v_t)| = |\mathcal{S}^+(A_j) \cup A_j| + |\mathcal{S}^+(A_k) \cup A_k| - 2|\mathcal{S}^+(A_t) \cup A_t| = |\mathcal{S}^+(A_j)| + 1 + |\mathcal{S}^+(A_j)| + 1 - 2(|\mathcal{S}^+(A_t)| + 1) = |\mathcal{S}^+(A_j)| + |\mathcal{S}^+(A_j)| - 2|\mathcal{S}^+(A_t)| = d_G^{\mathcal{V}}(V_{v_j}, V_{v_k})$. □

6.6.2 Jaccard's Distance

We can define a similarity measure between two families of sets based on elements and from this measure we can determine the distance between them. We adopt the **Jaccard's coefficient** [Jaccard, 1901] defining the similarity between two sets. Let A and B be two sets, then the Jaccard's coefficient is $J(A, B) = \frac{|A \cap B|}{|A \cup B|}$ and the Jaccard's distance is $d_J(A, B) = 1 - J(A, B)$; please see Definition 4.21 of Section 4.1.6.

Definition 6.9.
Let $\mathcal{A}_I$ and $\mathcal{B}_I$ be two families of sets, then their Jaccard's distance is defined to be:

$$d_J(\mathcal{A}, \mathcal{B}) = 1 - \frac{|\bigcup \mathcal{A}_I \cap \bigcup \mathcal{B}_I|}{|\bigcup \mathcal{A}_I \cup \bigcup \mathcal{B}_I|} \tag{6.50}$$

We define two families to be disjoint if their Jaccard's distance is equal to one.

In order to obtain the Jaccard's distance between two families $\mathcal{A}_I$ and $\mathcal{B}_J$ we have to calculate the intersection of all the elements in $\mathcal{A}_I$ with all the elements in $\mathcal{B}_J$; this operation can

be accomplished straightforwardly in the NS-M. Indeed, let $\mathcal{A}_I$ and $\mathcal{B}_J$ be two NS-F, then by the definition of NS-F we know that $\exists A \in \mathcal{A}_I \mid A = \bigcup \mathcal{A}_I$ and $\exists B \in \mathcal{B}_J \mid B = \bigcup \mathcal{B}_J$. So, the Jaccard's distance between $\mathcal{A}_I$ and $\mathcal{B}_J$ in the NS-M is defined to be:

$$d_J(\mathcal{A}, \mathcal{B}) = d_J(A, B) \tag{6.51}$$

On the other hand, in an INS-F does not exist a set that contains all the elements of the family. Let $\mathcal{A}_I$ and $\mathcal{B}_J$ be two INS-F such that $|\mathcal{A}_I| > n \in \mathbb{N}$ and $|\mathcal{B}_J| > m \in \mathbb{N}$. Let $\{A_1, \dots A_n\} \in \mathcal{A}_I$ be a collection of sets in $\mathcal{A}_I$ such that $\forall A_j \in \mathcal{A}_I, j \in [1, n] \Rightarrow \mathcal{S}^-(A_j) = 0$ and $\{B_1, \dots B_m\} \in \mathcal{B}_J$ be a collection of sets in $\mathcal{B}_J$ such that $\forall B_k \in \mathcal{B}_J, k \in [1, m] \Rightarrow \mathcal{S}^-(B_k) = 0$, then $\bigcup \mathcal{A}_I = \bigcup\{A_1, \dots, A_n\}$ and $\bigcup \mathcal{B}_J = \bigcup\{B_1, \dots, B_m\}$. So, we can say that:

$$d_J(\mathcal{A}, \mathcal{B}) = d_J(\cup\mathcal{A}_I, \cup\mathcal{B}_J) = d_J(\cup\{A_1, \dots, A_n\}, \cup\{B_1, \dots, B_m\}) \tag{6.52}$$

Alternatively, from the definition of INS-M we know that $\exists A_j \in \mathcal{A}_I \mid \forall A_k \in \mathcal{A}_I, k \neq j \Rightarrow A_j \subset A_k$ that is the common subset of the family. If we apply function ξ defined in Definition 6.4 to A_j (let B_k be the common subset of all the sets in $\mathcal{B}_J$) we obtain a set containing all the elements in the family $\mathcal{A}_I$ (respectively, $\xi(B_k)$ is the set containing all the elements in $\mathcal{B}_J$), then:

$$d_J(\mathcal{A}, \mathcal{B}) = d_J(\xi(A_j), \xi(B_k)) \tag{6.53}$$

We reduced the Jaccard's distance between two families of sets to the distance between two sets containing all the elements in the families – both NS-F and INS-F; in this way all the results proved for the Jaccard's distance between sets – please refer to Section 4.1.6 – are still valid when it is applied to families of sets. The Jaccard's distance is the same for two ordered or not-ordered families of sets; the order between the subsets does not change the measure of distance because it is based only on the element values and not on the structure of the families.

6.6.3 Structural Distance

In order to define the **structural distance between two families of sets** we need to take into account *partially-ordered families* of (ordered or not ordered) sets; i.e. poNS-F and poINS-F. First-of-all, we present the structural distance for a couple of poNS-F and then we shall extend it to the poINS-F. We define the structural distance measure to capture the structural similarity between two families of sets ignoring the element values. The structural distance can be seen as the counterpart of the Jaccard's distance. Indeed, the Jaccard's distance takes into account only the elements belonging to the families and not their organization into the nested sets; conversely, the structural distance takes into account only the relationships between the nested sets and not the elements belonging to them.

In order to define the structural distance between families we base our metric on the Parseval metric [Abney et al., 1991] and on the work on tree distance presented by Gallé in [Gallé, 2010]. These metrics are based on the concept of "brackets of tree" which is a set of intervals that permits us to reconstruct the tree from a set of values – please refer to Section 4.2.5

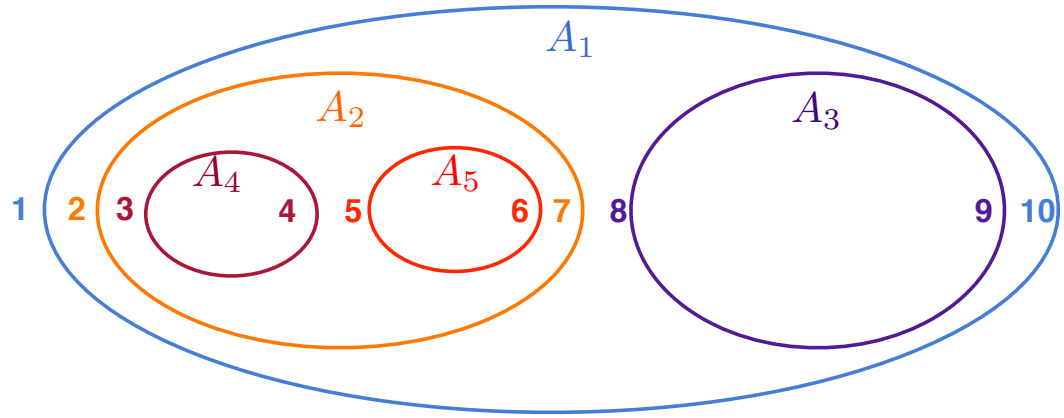


Figure 6.15: The integer encoding of a poNS-M.

for a detailed description of these distance measures. The definition of the structural distance measure starts from the definition of a set of integer intervals for each family of sets. We can choose between many hierarchical encodings proposed in the relational database literature for solving recursive queries in an algebraic way [Christophides et al., 2004; Härder et al., 2007; Tropashko, 2005; Xu et al., 2009]; anyway, the choice of the encoding is orthogonal to the metric, thus for sake of simplicity we pick up the basic integer encoding adopted by Celko [Celko, 2000]. The encoding of the families is the main difference between the definition of structural distance of two poNS-F and two poINS-F; indeed, as we will see, it is not possible to straightforwardly apply the integer encoding used for poNS-F to a poINS-F.

Accordingly to [Celko, 2000] a poNS-F can be encoded by a sequence of integer intervals assigned to each set. We recall the integer interval encoding by means of the following example.

Example 6.15. *Let $\mathcal{A}_{\langle I,<\rangle}$ be a poNS-F where $\mathcal{A}_{\langle I,<\rangle} = \{A_1, A_2, A_3, A_4, A_5\}$ such that $A_2 \subset A_1, A_3 \subset A_1, A_4 \subset A_2$ and $A_5 \subset A_2$. We can see a graphical representation of this family in Figure 6.15, where each set is assigned to an integer interval $[a, b]$ where $a, b \in \mathbb{N}$. Each interval identifies in a unique way a set within the family; the integer values are assigned proceeding left to right starting from 1.*

We can see that $A_1 = [1, 10], A_2 = [2, 7], A_3 = [8, 9], A_4 = [3, 4]$ and $A_5 = [5, 6]$.

Following this encoding we can see that each set is assigned to an interval $[a, b]$ where $a, b \in \mathbb{N} \wedge a < b$. For each po-family $\mathcal{A}_{\langle I,<\rangle}$ we define a poset $\langle W_{\mathcal{A}}, <\rangle$ where each element is an integer interval encoding a set in $\mathcal{A}_I$. In the example above $W_{\mathcal{A}} = ([1, 10], [2, 7], [8, 9], [3, 4], [5, 6])$. We need $W_{\mathcal{A}}$ to be unique for $\mathcal{A}_{\langle I,<\rangle}$ and for this reason the encoded family must be partially ordered; otherwise, the interval associated with each set can change with the relative position of the subsets. We call w_i the i^{th} element of $W_{\mathcal{A}}$ where $w_i = [a_i, b_i]$[2]; then, for each $w_i, w_j \in W_{\mathcal{A}}$,

[2]Please note that if there is any risk of ambiguity we shall use the full notation comprising the apexes which indicate the family we are taking into account: $w_i^{\mathcal{A}} = [a_i^{\mathcal{A}}, b_i^{\mathcal{A}}]$.

$w_i < w_j$ if and only if $a_i < a_j$. In order to define the structural distance we take into account the sets in $\mathcal{A}_{\langle I,<\rangle}$ which do not have any subsets thus every $A_j \in \mathcal{A}_{\langle I,<\rangle} \mid \mathcal{S}^+(A_j) = 0$; if we map this family into a tree, the sets we are considering correspond to the external nodes of the tree. If we consider these sets from the encoding point-of-view, they are associated with an interval, say $w_j = [a_j, b_j]$, such that $b_j - a_j = 1$; we call this interval the *1-interval*.

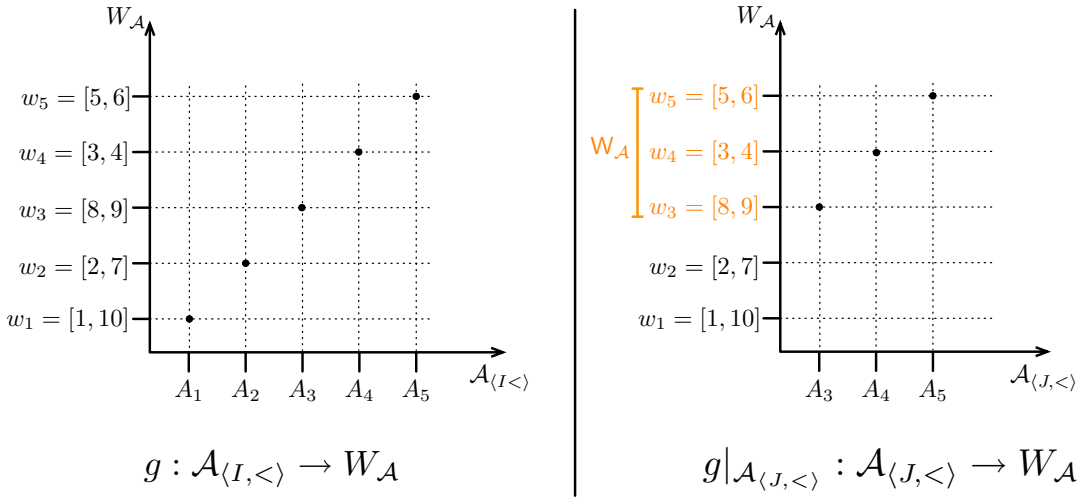


Figure 6.16: The function g mapping the sets of the family $\mathcal{A}_{\langle I,<\rangle}$ (Example 6.15) to the integer intervals encoding them and its restriction to the subfamily $\mathcal{A}_{\langle J,<\rangle}$.

Formally, we define a bijective function $g : \mathcal{A}_{\langle I,<\rangle} \rightarrow W_{\mathcal{A}}$ where $\mathcal{A}_{\langle I,<\rangle}$ is a NS-F and $W_{\mathcal{A}}$ a poset containing intervals defined by the integer encoding [Celko, 2000]. $W_{\mathcal{A}}$ is ordered then $w_i < w_j < w_k \Rightarrow a_i < a_j < a_k$. We define $\mathcal{A}_{\langle J,<\rangle}$ to be a subfamily (poNS-F) of $\mathcal{A}_{\langle I,<\rangle}$ ($\mathcal{A}_{\langle J,<\rangle} \subset \mathcal{A}_{\langle I,<\rangle}$) such that $A_k \in \mathcal{A}_{\langle J,<\rangle}$ if and only if $A_k \in \mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{S}^+(A_k) = 0$. Then, we define with $g|_{\mathcal{A}_{\langle J,<\rangle}}$ the restriction of g to $\mathcal{A}_{\langle J,<\rangle}$ that associates the sets without any subsets in $\mathcal{A}_{\langle I,<\rangle}$ to an interval in $W_{\mathcal{A}}$. Let us call $\mathsf{W}_{\mathcal{A}}$ the range of $g|_{\mathcal{A}_{\langle J,<\rangle}}$ where $\mathsf{W}_{\mathcal{A}} \subset W_{\mathcal{A}}$ and $w_i \in \mathsf{W}_{\mathcal{A}}$ if and only if $w_i \in W_{\mathcal{A}}$ and $b_i - a_i = 1$.

In Figure 6.16 we can see a graphical representation of the function g and its restriction to $\mathcal{A}_{\langle J,<\rangle}$ in the case of Example 6.15. We call $\mathsf{W}_{\mathcal{A}}$ the **1-interval poset** of $\mathcal{A}_{\langle I,<\rangle}$. The poset $\mathsf{W}_{\mathcal{A}}$ characterizes the structure of the family $\mathcal{A}_{\langle I,<\rangle}$, indeed from $\mathsf{W}_{\mathcal{A}}$ we can calculate the cardinality of $\mathcal{A}_{\langle I,<\rangle}$ and infer the relationships between the sets. The 1-interval poset contains enough information to calculate the structural distance between two families of subsets.

In order to be able to compare the 1-intervals of two poNS-F $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle K,<\rangle}$ with 1-interval posets respectively W_A and W_D, we suppose an **assignment function** $f : \mathsf{W}_A \rightarrow \mathsf{W}_D \cup \{w_0\}$ where $w_0 = [0,0]$; please note that the interval $[0,0]$ cannot be assigned to any set by the integer encoding [Celko, 2000]. The assignment function f is required to be injective, except possibly for $[0,0]$. The role of $[0,0]$ in the image of f is to permit us to assign intervals from $\mathsf{W}_{\mathcal{A}}$ which otherwise would not be assigned.

We can define several different assignment functions; for sake of simplicity we define f such that it assigns an element $w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}$ to an element $w_x^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}$ if and only if $a_i^{\mathcal{A}} =$

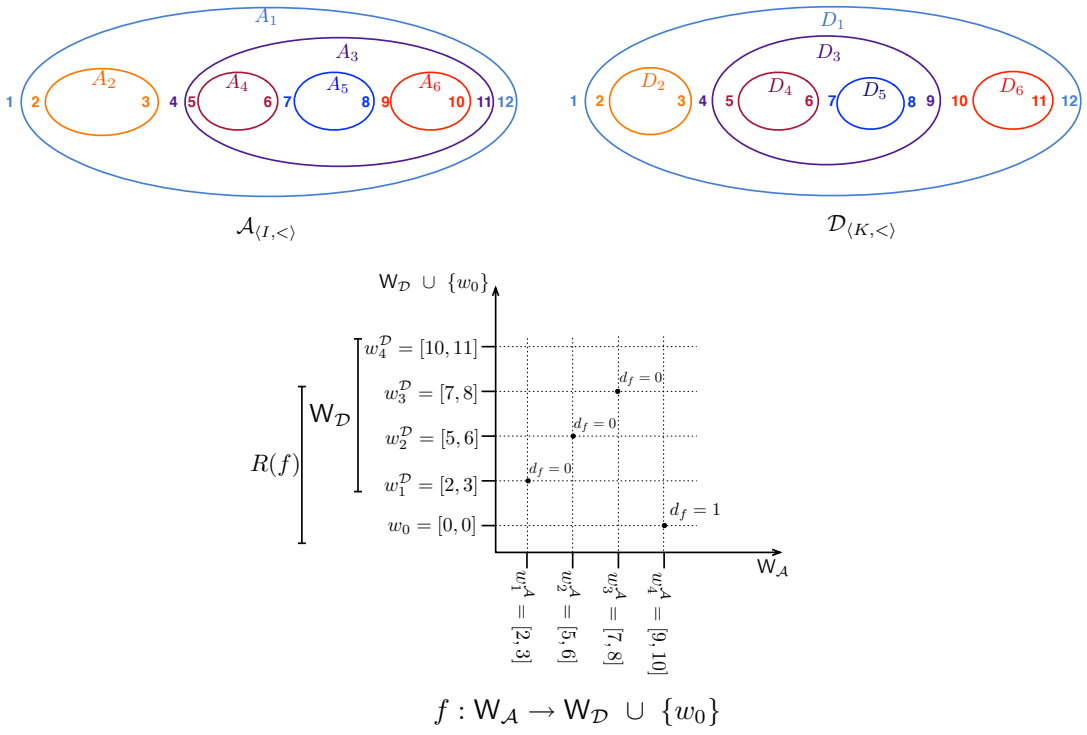


Figure 6.17: Two poNS-F encoded by integer intervals and an assigning function f.

$a_x^\mathcal{D} \wedge b_i^\mathcal{A} = b_x^\mathcal{D}$, otherwise $w_i^\mathcal{A}$ is assigned to $w_0 = [0, 0]$ (call it *null assignment*).

We define the **binary assignment function** d_f associated with the assignment function f as:

$$d_f(w_i, f(w_i)) = \begin{cases} 1 & \text{if } f(w_i) = \{w_0\} = [0, 0] \\ 0 & \text{otherwise.} \end{cases} \tag{6.54}$$

Let $R(f)$ denotes the range of the assignment function $f : \mathsf{W}_A \rightarrow \mathsf{W}_D \cup \{w_0\}$; i.e. $R(f)$ is a set containing the elements of $\mathsf{W}_\mathcal{D}$ matched by an element in the domain $\mathsf{W}_\mathcal{A}$ of the function f. Note that $R(f)$ may be the whole of $\mathsf{W}_D \cup \{w_0\}$ or a proper subset of $\mathsf{W}_D \cup \{w_0\}$ [Copson, 1968]. Let us consider an example.

Example 6.16. *Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle K,<\rangle}$ be two poNS-F such that $\mathcal{A}_{\langle I,<\rangle} = \{A_1, A_2, A_3, A_4, A_5, A_6\}$ and $\mathcal{D}_{\langle K,<\rangle} = \{D_1, D_2, D_3, D_4, D_5, D_6\}$. In Figure 6.17 we represent these poNS-F by means of Eulero-Venn diagrams which show the relationships between the sets.*

In this figure we also show the integer intervals assigned to the sets and the assigning function f presented above.

Now, we can define the *structural distance* between two poNS-F.

Definition 6.10.

Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle K,<\rangle}$ be two poNS-F, $\mathsf{W}_\mathcal{A}$ and $\mathsf{W}_\mathcal{D}$ be the associated 1-interval posets, f :

$\mathsf{W}_{\mathcal{A}} \to \mathsf{W}_{\mathcal{D}} \cup \{w_0\}$ *be the assignment function,* $R(f)$ *be the range of the assigning function, and* $d_f(w_i, f(w_i))$ *be the binary assignment function. Then,*

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(f)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} \tag{6.55}$$

is defined to be the ***structural distance*** *between two poNS-F.*

The equation 6.55 gives a penalty of a maximal distance for every integer interval in $\mathsf{W}_{\mathcal{D}}$ to which no interval in $\mathsf{W}_{\mathcal{A}}$ is assigned.

Example 6.17. *If we consider Example 6.16 the structural distance between* $\mathcal{A}_{\langle I,<\rangle}$ *and* $\mathcal{D}_{\langle K,<\rangle}$ *is*

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(f)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} = \frac{1+1}{8} = 0.25$$

It is important to prove that the defined structural distance is a proper metric for the partially-ordered families of nested sets; thus, we have to prove four main properties of the distance: identity, non-negativity, symmetry and the triangle inequality. We introduce a proposition and a corollary with some intermediate results that will be useful when proving that the structural distance is a proper metric.

Proposition 6.22. *Let* $\mathcal{A}_{\langle I,<\rangle}$ *and* $\mathcal{D}_{\langle J,<\rangle}$ *be two poNS-F, and* $d_S(\mathcal{A}, \mathcal{D})$ *be the structural distance between them. Then,*

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(f)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} = \frac{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}| - 2|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}}|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} \tag{6.56}$$

Proof:

Let us consider the meaning of $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}}))$*; this sum adds a unit for each* $w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid f(w_i^{\mathcal{A}}) \neq w_i^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}$*. This means that if there are no common elements between* $\mathsf{W}_{\mathcal{A}}$ *and* $\mathsf{W}_{\mathcal{D}} \Rightarrow \mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}} = \emptyset$ *then* $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) = |\mathsf{W}_{\mathcal{A}}|$*. Otherwise, we have to subtract the intersection and then in the general case* $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) = |\mathsf{W}_{\mathcal{A}}| - |\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}}|$.

$|\mathsf{W}_{\mathcal{D}} \setminus R(f)|$ *adds a unit for every element of* $\mathsf{W}_{\mathcal{D}}$ *that is not matched by an element of* $\mathsf{W}_{\mathcal{A}}$ *thus* $|\mathsf{W}_{\mathcal{D}} \setminus R(f)| = |\mathsf{W}_{\mathcal{D}}| - |\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}}|$*. The reader may notice that the element* w_0 *it is not taken into account in this last equation; it is important to underline that* $w_0 \notin \mathsf{W}_{\mathcal{D}} \Rightarrow w_0 \notin \mathsf{W}_{\mathcal{D}} \setminus R(f)$*, thus when we consider the cardinality of* $\mathsf{W}_{\mathcal{D}} \setminus R(f)$*, the element* w_0 *is not taken into account.*

So, $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(f)| = |\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}| - 2|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}}|$. □

The following corollary defines the upper and lower bounds of the members of the structural distance and it is a direct consequence of Proposition 6.22.

Corollary 6.23. *Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ be two poNS-F, $\mathsf{W}_{\mathcal{A}}$ and $\mathsf{W}_{\mathcal{D}}$ be their associated 1-interval posets and $f : \mathsf{W}_{\mathcal{A}} \to \mathsf{W}_{\mathcal{D}} \cup \{w_0\}$ be the assignment function, and d_f the associated binary assignment function. Then:*

$$0 \leq \sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) \leq |\mathsf{W}_{\mathcal{A}}| \tag{6.57}$$

$$0 \leq |\mathsf{W}_{\mathcal{D}} \setminus R(f)| \leq |\mathsf{W}_{\mathcal{D}}| \tag{6.58}$$

Proof:
The prove of these two properties can be trivially derived from the proof of the Proposition 6.22.□

Lemma 6.24. *Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ be two poNS-F. Then, the structural distance $d_S(\mathcal{A}, \mathcal{D})$ is a **proper metric**.*

Proof:
Let $\mathcal{A}_{\langle I,<\rangle}$, $\mathcal{D}_{\langle J,<\rangle}$ and $\mathcal{O}_{\langle K,<\rangle}$ be three poNS-F, $\mathsf{W}_{\mathcal{A}}$, $\mathsf{W}_{\mathcal{D}}$ and $\mathsf{W}_{\mathcal{O}}$ be their associated 1-interval posets and $f : \mathsf{W}_{\mathcal{A}} \to \mathsf{W}_{\mathcal{O}} \cup \{w_0\}$, $g : \mathsf{W}_{\mathcal{D}} \to \mathsf{W}_{\mathcal{O}} \cup \{w_0\}$ and $h : \mathsf{W}_{\mathcal{A}} \to \mathsf{W}_{\mathcal{D}} \cup \{w_0\}$ be the assignment functions associated respectively with the binary assignment functions d_f, d_g and d_h.

Let

$$d_S(\mathcal{A}, \mathcal{O}) = \frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_f(w_i^{\mathcal{A}}, f(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{O}} \setminus R(f)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{O}}|},$$

$$d_S(\mathcal{D}, \mathcal{O}) = \frac{\sum_{w_i^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}} d_g(w_i^{\mathcal{D}}, g(w_i^{\mathcal{D}})) + |\mathsf{W}_{\mathcal{O}} \setminus R(g)|}{|\mathsf{W}_{\mathcal{D}}| + |\mathsf{W}_{\mathcal{O}}|}, \text{ and}$$

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(h)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|}$$

be the structural distances between $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{O}_{\langle K,<\rangle}$, $\mathcal{D}_{\langle J,<\rangle}$ and $\mathcal{O}_{\langle K,<\rangle}$ and $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ respectively.

In order to prove that d_S is a proper metric we have to show:

Identity: $d_S(\mathcal{A}, \mathcal{A}) = 0$

Non-negativity: $d_S(\mathcal{A}, \mathcal{D}) \geq 0$

Symmetry: $d_S(\mathcal{A}, \mathcal{D}) = d_S(\mathcal{D}, \mathcal{A})$

Triangle Inequality: $d_S(\mathcal{A}, \mathcal{O}) + d_S(\mathcal{D}, \mathcal{O}) \geq d_S(\mathcal{A}, \mathcal{D})$

*Let us prove identity and non-negativity altogether. We know that $|\mathsf{W}_{\mathcal{A}}| \geqslant 0$ and $|\mathsf{W}_{\mathcal{D}}| \geqslant 0$ then if $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(h)| = 0 \Rightarrow \forall\, w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}, \exists w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}} \mid w_i^{\mathcal{A}} = w_j^{\mathcal{D}} \Rightarrow R(h) = \mathsf{W}_{\mathcal{D}} \Rightarrow \mathsf{W}_{\mathcal{A}} = \mathsf{W}_{\mathcal{D}} \Rightarrow \mathcal{D} = \mathcal{A} \Rightarrow d_S(\mathcal{A}, \mathcal{A}) = d_S(\mathcal{D}, \mathcal{D}) = 0$ (**identity**).*

If $\mathcal{A} \neq \mathcal{D} \Rightarrow \exists w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid h(w_i^{\mathcal{A}}) = \{w_0\} = [0,0] \Rightarrow \sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(h)| > 0$. *From Corollary 6.23 we know that* $|\mathsf{W}_{\mathcal{D}} \setminus R(h)| \geqslant 0$ *then* $\frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(h)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} > 0$ (***non-negativity***).

In order to prove ***symmetry*** *let us define the distance between* $\mathcal{D}$ *and* $\mathcal{A}$ *as*

$$d_S(\mathcal{D}, \mathcal{A}) = \frac{\sum_{w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}} d_l(w_j^{\mathcal{D}}, l(w_j^{\mathcal{D}})) + |\mathsf{W}_{\mathcal{A}} \setminus R(l)|}{|\mathsf{W}_{\mathcal{D}}| + |\mathsf{W}_{\mathcal{A}}|}$$

where $l : \mathsf{W}_D \to \mathsf{W}_{\mathcal{A}} \cup \{w_0\}$ *is the usual assignment function and* d_l *its binary assignment function.*

$$d_S(\mathcal{A}, \mathcal{D}) = d_S(\mathcal{D}, \mathcal{A}) \Rightarrow$$

$$\frac{\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) + |\mathsf{W}_{\mathcal{D}} \setminus R(h)|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_{\mathcal{D}}|} = \frac{\sum_{w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}} d_l(w_j^{\mathcal{D}}, l(w_j^{\mathcal{D}})) + |\mathsf{W}_{\mathcal{A}} \setminus R(l)|}{|\mathsf{W}_{\mathcal{D}}| + |\mathsf{W}_{\mathcal{A}}|}$$

$\forall\, w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}})) = 1 \Rightarrow \nexists w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}} \mid w_j^{\mathcal{D}} = w_i^{\mathcal{A}} \Rightarrow \exists\, w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}} \mid w_j^{\mathcal{D}} \notin R(h) \Rightarrow \nexists\, w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid h(w_i^{\mathcal{A}}) = w_j^{\mathcal{D}} \Rightarrow \nexists\, w_j^{\mathcal{D}} \mid l(w_j^{\mathcal{D}}) = w_i^{\mathcal{A}} \Rightarrow \exists\, w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid w_i^{\mathcal{A}} \in (\mathsf{W}_{\mathcal{A}} \setminus R(l))$. $\forall w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}} \mid h(w_i^{\mathcal{A}}) = w_0 \Rightarrow w_i^{\mathcal{A}} \in (\mathsf{W}_{\mathcal{A}} \setminus R(l))$.

This means that for every unit added to $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(w_i^{\mathcal{A}}, h(w_i^{\mathcal{A}}))$ *there is a correspondent element belonging to* $\mathsf{W}_{\mathcal{A}} \setminus R(l)$, *thus* $\sum_{w_i^{\mathcal{A}} \in \mathsf{W}_{\mathcal{A}}} d_h(e_i^{\mathcal{A}}, h(e_i^{\mathcal{A}})) = |\mathsf{W}_{\mathcal{A}} \setminus R(l)|$. *Symmetrically,* $|\mathsf{W}_{\mathcal{D}} \setminus R(h)| = \sum_{w_j^{\mathcal{D}} \in \mathsf{W}_{\mathcal{D}}} d_l(w_j^{\mathcal{D}}, l(w_j^{\mathcal{D}}))$, *for the same reasons. So* $d_S(\mathcal{A}, \mathcal{D}) = d_S(\mathcal{D}, \mathcal{A})$ *(symmetry).*

Now, we have to prove the ***triangle inequality****:* $d_S(\mathcal{A}, O) + d_S(\mathcal{D}, O) \geq d_S(\mathcal{A}, \mathcal{D})$. *We know that the maximum distance between two families is equal to 1; so, if we consider* $\mathcal{A}$ *and* $\mathcal{D}$ *their maximal structural distance is* $d_S(\mathcal{A}, \mathcal{D}) = 1 \Rightarrow \mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}} = \emptyset$. *Then, if*

$$d_S(\mathcal{A}, O) + d_S(\mathcal{D}, O) \geq 1$$

is proved to be true, the same inequality is true for any value of $d_S(\mathcal{A}, \mathcal{D}) < 1$. *We prove the triangle inequality by induction on the cardinality of* W_O. *Please note that we have maximized the distance between* $\mathcal{A}$ *and* $\mathcal{D}$ *so:* $\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_O = \emptyset$.

Base $|\mathsf{W}_O| = 1$. *We point out two base cases:*

Base Case 1. $|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_O| = 0 \wedge |\mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_O| = 0 \Rightarrow d_S(\mathcal{A}, O) = 1 \wedge d_S(\mathcal{D}, O) = 1 \Rightarrow d_S(\mathcal{A}, O) + d_S(\mathcal{D}, O) = 2 > 1$.

Base Case 2. $|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_O| = 1 \vee |\mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_O| = 1$. *If* $d(\mathcal{A}, O) = 1$ *then* $d(\mathcal{D}, O) = 0$ *or vice versa.*

Inductive hypothesis: the triangle inequality is verified for $|\mathsf{W}_O| = n$.

$$d_S(\mathcal{A}, O) + d_S(\mathcal{D}, O) \geq 1 \Rightarrow \frac{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_O| - 2|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_O|}{|\mathsf{W}_{\mathcal{A}}| + |\mathsf{W}_O|} + \frac{|\mathsf{W}_{\mathcal{D}}| + |\mathsf{W}_O| - 2|\mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_O|}{|\mathsf{W}_{\mathcal{D}}| + |\mathsf{W}_O|} \geqslant 1 \tag{6.59}$$

In order to make the notation of the proof more readable we do the following substitutions: $|\mathsf{W}_{\mathcal{A}}| = a$, $|\mathsf{W}_{\mathcal{D}}| = d$, $|\mathsf{W}_{O}| = o$, $|\mathsf{W}_{\mathcal{A}} \cap \mathsf{W}_{O}| = \alpha$ *and* $|\mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_{O}| = \beta$. *Then, we can rewrite equation 6.59 as:*

$$\frac{a + o - 2\alpha}{a + o} + \frac{d + o - 2\beta}{d + o} \geqslant 1 \tag{6.60}$$

Then we can do other two substitutions: $x = a + o - 2\alpha$ *and* $y = d + o - 2\beta$ *and rewrite equation 6.60 as it follows obtaining a result that will be used to prove the inductive step:*

$$\frac{x}{a + o} + \frac{y}{d + o} \geqslant 1 \Rightarrow ay + oy + xd + xo \geqslant ad + ao + o^2 + od \tag{6.61}$$

Another useful result comes out from the following equation:

$$\begin{aligned}\frac{a + o - 2\alpha}{a + o} + \frac{d + o - 2\beta}{d + o} \geqslant 1 \Rightarrow & (d + o)(a + o - 2\alpha) + (a + o)(d + o - 2\beta) \geqslant (a + o)(d + o) \\ & \Rightarrow d(-2\alpha + a) + o(-2\alpha + a) + o(o + d) - 2\beta(a + o) \geqslant 0 \\ & \Rightarrow -2\alpha + a + o - 2\beta\Big(\frac{a + o}{d + o}\Big) \\ & \Rightarrow a + o \geqslant 2\alpha + 2\beta\Big(\frac{a + o}{d + o}\Big)\end{aligned} \tag{6.62}$$

Inductive step: $|\mathsf{W}_{O}| = n + 1$. *We know that the triangle inequality is verified for* $|\mathsf{W}_{O}| = n$, *so we add an element* w^{O}_{n+1} *to* W_{O}. *We have now to prove two possible cases:*

A. $w^{O}_{n+1} \in \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{A}} \Rightarrow w^{O}_{n+1} \notin \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{D}} \vee w^{O}_{n+1} \in \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{D}} \Rightarrow w^{O}_{n+1} \notin \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{A}}$.

B. $w^{O}_{n+1} \notin \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{A}} \wedge w^{O}_{n+1} \notin \mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_{\mathcal{D}}$.

Let us prove the case **A** *for* $w^{O}_{n+1} \in \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{A}}$ *(the proof for* $w^{O}_{n+1} \in \mathsf{W}_{\mathcal{D}} \cap \mathsf{W}_{O}$ *is symmetric to this one).*

Starting from the triangle inequality with substituted notation we know that for $|\mathsf{W}_{O}| = n$ *the following equation is true:* $\frac{x}{a+o} + \frac{y}{d+o} \geqslant 1$. *For* $w^{O}_{n+1} \in \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{A}}$ *this equation becomes:*

$$\frac{x}{a + o + 1} + \frac{y + 1}{d + o + 1} \geqslant 1$$

because the $d_S(\mathcal{A}, O)$ *diminishes of a little quantity determined by the augmentation of the dimension of o in the denominator. x does not augment because we add an element to* W_{O} *but it is matched by an element in* $\mathsf{W}_{\mathcal{A}}$ *so it does not count in the numerator of the fraction defining the structural distance. On the other hand, the newly added element is not matched in* $\mathsf{W}_{\mathcal{D}}$ *because* $w^{O}_{n+1} \notin \mathsf{W}_{O} \cap \mathsf{W}_{\mathcal{D}}$ *by definition. This means that the numerator y is augmented of a unit in the* $n + 1$ *step. The denominator of the second member behaves like the denominator in the first member of the equation.*

Then:

$$\frac{x}{a + o + 1} + \frac{y + 1}{d + o + 1} \geqslant 1 \Rightarrow$$

$$xd + xo + x + ay + a + oy + o + y + 1 \geqslant ad + ao + od + o^2 + a + o + d + o + 1$$

From the inductive hypothesis we know that for $|\mathsf{W}_O| = n$ *the equation* $ay + oy + xd + xo \geqslant ad + ao + o^2 + od$ *is verified. Then, we know that this inequality is true also for the* $n + 1$ *case because in this step we did not modify the dimensions of* o *and* y *but we rewrite the triangle inequality by adding a unit to the denominators and to the numerator of the second member of the inequality. So, it remains to prove that:*

$$x + a + o + y + 1 \geq 2o + d + 1 + a \Rightarrow x + y \geq o + d \tag{6.63}$$

If $x + y \geq o + d$ *is true for* $|\mathsf{W}_O| = n$ *then it is also true for* $|\mathsf{W}_O| = n + 1$*. So,* $x + y \geq o + d \Rightarrow a + o \geq 2\alpha + 2\beta$*. By the inductive hypothesis we know that* $a + o \geqslant 2\alpha + 2\beta\left(\frac{a+o}{d+o}\right)$*; so, if* $a \geq d$ *the case is verified.*

Now, we have to prove the case in which $a < d$*; to prove this part we take another approach. In order to simplify the notation we call the structural distance between two families in the step* n*:* $d_n(\mathcal{D}, \mathcal{O})$ *and the correspondent one at the step* $n + 1$*:* $d_{n+1}(\mathcal{D}, \mathcal{O})$*. Then we have to prove that:*

$$d_{n+1}(\mathcal{A}, \mathcal{O}) + d_{n+1}(\mathcal{D}, \mathcal{O}) \geq 1.$$

We can rewrite this inequality in the following way:

$$d_n(\mathcal{A}, \mathcal{O}) - \frac{d_n(\mathcal{A}, \mathcal{O})}{|\mathsf{W}_\mathcal{A}| + |\mathsf{W}_O| + 1} + d_n(\mathcal{D}, \mathcal{O}) + \frac{1}{|\mathsf{W}_\mathcal{D}| + |\mathsf{W}_O| + 1} \geq 1$$

The sum of the two distances is greater than one in the n step, so to become < 1 *in the* $n + 1$*, the minus term* $\frac{d_n(\mathcal{A},\mathcal{O})}{|\mathsf{W}_\mathcal{A}|+|\mathsf{W}_O|+1}$ *must diminish at a bigger rate that the growing rate of the plus term* $\frac{1}{|\mathsf{W}_\mathcal{D}|+|\mathsf{W}_O|+1} \geq 1$*. By the initial conditions we know that* $d_n(\mathcal{A}, \mathcal{O}) < 1$*,* $|\mathsf{W}_\mathcal{D}| > |\mathsf{W}_\mathcal{A}|$ *and than* $|\mathsf{W}_O| >= 3$*. If we set all the values to their allowed minimum we have:* $|\mathsf{W}_\mathcal{A}| = 2$ *(*$\mathsf{W}_\mathcal{A}$ *must have an element in common with* W_O *and the distance between* $\mathcal{A}$ *and* O *must be* < 1*),* $|\mathsf{W}_\mathcal{D}| = 3$ *(because it must be bigger than* $\mathsf{W}_\mathcal{A}$*). We cannot set these values to be smaller than these values. Then, if the inequality is verified for these values, it must be true also for bigger allowed values. Vice versa, we set* $d_n(\mathcal{A}, \mathcal{O})$ *to the maximum allowed value that is* $3/5 = 0.6$*. Then we can consider an integer variable* $k \in [1, m]$ *with* $m \in \mathbb{N}$*. We obtain the following functions:*

$$b_1(k) = \frac{d_n(\mathcal{A}, \mathcal{O})}{|\mathsf{W}_\mathcal{A}| + |\mathsf{W}_O| + k} = \frac{0.6}{5 + k}$$

and

$$b_2(k) = \frac{k}{|\mathsf{W}_\mathcal{D}| + |\mathsf{W}_O| + k} = \frac{k}{6 + k}$$

where k grows when the cardinality of W_O grows and every new element added to W_O belongs also to $\mathsf{W}_\mathcal{A}$ because of the initial conditions. We can see that $b_2(k) - b_1(k) < 0$ for no $k \geqslant 1$.

Then, for $k \geq 1$ where $k \in \mathbb{N}$, $b_2(k) - b_1(k)$ is never a negative quantity. Augmenting the cardinality of the sets and the distance between $\mathcal{A}$ and O the $b_2(k) - b_1(k)$ becomes closer and closer to zero but it does not change its sign. Thus:

$$\frac{1}{|\mathsf{W}_\mathcal{D}| + |\mathsf{W}_O| + 1} - \frac{d_n(\mathcal{A}, O)}{|\mathsf{W}_\mathcal{A}| + |\mathsf{W}_O| + 1} \geqslant 0$$

The proof of the case in which $w^O_{n+1} \in \mathsf{W}_O \cap \mathsf{W}_\mathcal{D}$ is proved straightforwardly following this proof.

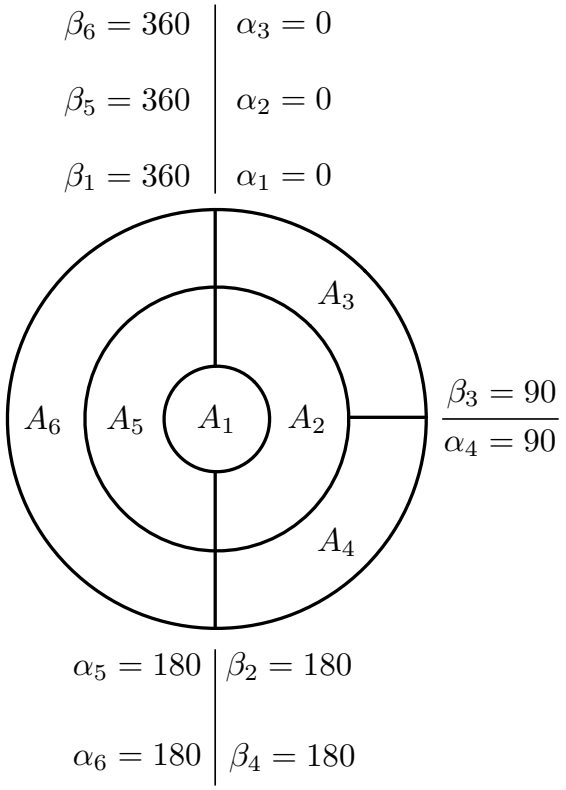
*Now, we have prove the case **B.** in which: $w^O_{n+1} \notin \mathsf{W}_O \cap \mathsf{W}_\mathcal{A} \wedge w^O_{n+1} \notin \mathsf{W}_\mathcal{D} \cap \mathsf{W}_\mathcal{D}$. We have to prove that $d_{n+1}(\mathcal{A}, O) + d_{n+1}(\mathcal{D}, O) \geq 1$ and under the conditions of this case we can rewrite it as:*

$$d_n(\mathcal{A}, O) + \frac{1}{|\mathsf{W}_\mathcal{A}| + |\mathsf{W}_O| + 1} + d_n(\mathcal{D}, O) + \frac{1}{|\mathsf{W}_\mathcal{D}| + |\mathsf{W}_O| + 1} \geq 1$$

We can see that both the members in the left part of the equation are augmented by a positive quantity, then the triangle inequality is always verified. □

We have defined the structural distance between to poNS-F and we have shown that this measure is a proper metric. In order to adopt this metric with poINS-F we have to define a new encoding for the sets in a poINS-F. Basically, we have to define the interval poset $W_\mathcal{A}$ for $\mathcal{A}_{\{I,<\}}$ when it is a poINS-F. In this case, we cannot exploit well-known node labeling techniques defined for trees like we have done for the NS-M, because the INS-M reverses the logic of the tree making those node labeling techniques inapplicable to the poINS-F. A naive approach is mapping the poINS-F into their corresponding poNS-F and calculate the structural distance between them. This approach gives the measure of the distance between two poINS-F passing through their representation in the NS-M. On the other hand, our aim is to define the NS-M and the INS-M as two interrelated but independent set data models. To accomplish this goal we define a proper encoding for the sets in a poINS-F exploiting the encoding presented in [Vegas et al., 2007] to identify a circular section in a DocBall. It is important to remember that, as we have pointed out in the beginning of this section, the choice of a particular encoding is orthogonal to the definition of the metric.

We can encode a set A_j of an poINS-F $\mathcal{A}_{\langle I,<\rangle}$ using its DocBall representation as guideline (as we have done with the Eulero-Venn diagrams for the poNS-F) by means of a triple of values $t_j = \langle l_j, \alpha_j, \beta_j \rangle$ where $l_j \in [1, n], n \in \mathbb{N}$ is the level of A_j in the DocBall, $\alpha_j \in [0, 360]$ is the starting angle of the section representing A_j in the DocBall and $\beta_j \in [0, 360]$ is the ending angle of the section representing A_j in the DocBall. The poINS-F $\mathcal{A}_{\langle I,<\rangle}$ is associated with a poset $\langle T_\mathcal{A}, <\rangle = \{t_1, \ldots, t_{|\mathcal{A}|}\}$ that we call the *triple encoding poset* of $\mathcal{A}_{\langle I,<\rangle}$. Let us consider



$$t_1 = \langle l_1, \alpha_1, \beta_1 \rangle = \langle 1, 0, 360 \rangle$$
$$t_2 = \langle l_2, \alpha_2, \beta_2 \rangle = \langle 2, 0, 180 \rangle$$
$$t_3 = \langle l_3, \alpha_3, \beta_3 \rangle = \langle 3, 0, 90 \rangle$$
$$t_4 = \langle l_4, \alpha_4, \beta_4 \rangle = \langle 3, 90, 180 \rangle$$
$$t_5 = \langle l_5, \alpha_5, \beta_5 \rangle = \langle 2, 180, 360 \rangle$$
$$t_6 = \langle l_6, \alpha_6, \beta_6 \rangle = \langle 3, 180, 360 \rangle$$

$$T_{\mathcal{A}} = \{t_1, t_2, t_3, t_4, t_5, t_6\}$$

Figure 6.18: A poINS-F represented by means of a DocBall and the definition of the encoding of each set in the poINS-F.

two triples $\{t_1, t_2\} \in \langle T_{\mathcal{A},<} \rangle$, then we say that $t_1 < t_2$ if and only if $l_1 = l_2 \wedge \beta_1 \leq \alpha_2$; two triples ($\{t_3, t_4\} \in \langle T_{\mathcal{A}}, < \rangle$) such that $l_3 \neq l_4$ are defined to be incomparable ($t_3 \| t_4$).

In Figure 6.18 we can see how $\langle T_{\mathcal{A}}, < \rangle$ is defined; every set is represented in the DocBall as a section and each section has a starting angle α and an ending angle β. These angles do not suffice to uniquely identify a section, thus we consider also the level of the sections; the inner circle has level 1 and so on and so forth.

In order to calculate the structural distance between two poINS-F $\mathcal{A}_{\langle I,< \rangle}$ and $\mathcal{D}_{\langle J,< \rangle}$ we lever on their associated triple encoding posets, respectively $\langle T_{\mathcal{A}}, < \rangle$ and $\langle T_{\mathcal{D}}, < \rangle$. As we have done for the NS-M, we define an assignment function $f : \langle T_{\mathcal{A}}, < \rangle \rightarrow \langle T_{\mathcal{D}}, < \rangle \cup \{t_0\}$ where $t_0 = \langle 0, 0, 0 \rangle$. Please note that the triple $\langle 0, 0, 0 \rangle$ cannot be assigned to any set by the defined encoding. This function is required to be injective, except possibly for $t_0 = \langle 0, 0, 0 \rangle$. The role of $\langle 0, 0, 0 \rangle$ in the image of f is to permit us to assign intervals from $\langle T_{\mathcal{A}}, < \rangle$ which otherwise would not be assigned. Then, the assignment function f is defined as follows: every $t_k^{\mathcal{A}} \in \langle T_{\mathcal{A}}, < \rangle$ is mapped into an element $t_h^{\mathcal{D}} \in \langle T_{\mathcal{D}}, < \rangle \cup t_0$ such that $f(t_k^{\mathcal{A}}) = t_h^{\mathcal{D}} \in \langle T_{\mathcal{D}}, < \rangle$ if $t_k^{\mathcal{A}} = t_h^{\mathcal{D}}$ or $f(t_k^{\mathcal{A}}) = t_0 \in \langle T_{\mathcal{D}}, < \rangle$ otherwise.

Now, we can define the binary assignment function d_f for the INS-M as:

$$d_f(t_i, f(t_i)) = \begin{cases} 1 & \text{if } f(t_i) = \{t_0\} = \langle 0,0,0\rangle \\ 0 & \text{otherwise.} \end{cases} \tag{6.64}$$

So, the binary assignment function is defined in the same way for the NS-M and the INS-M, thus the only difference between the two is the encoding of the sets. In NS-M we compare 1-interval posets encoding the families; in INS-M we compare posets containing triples. Thus, the structural distance d_S between two poINS-F is:

Definition 6.11.
Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle K,<\rangle}$ be two poINS-F, $T_{\mathcal{A}}$ and $T_{\mathcal{D}}$ be the associated triple encoding posets, $f : T_{\mathcal{A}} \to T_{\mathcal{D}} \cup \{t_0\}$ be the assignment function, $R(f)$ be the range of the assigning function, and $d_f(t_i, t(w_i))$ be the binary assigning function. Then,

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{t_i^{\mathcal{A}} \in T_{\mathcal{A}}} d_f(t_i^{\mathcal{A}}, f(t_i^{\mathcal{A}})) + |T_{\mathcal{D}} \setminus R(f)|}{|T_{\mathcal{A}}| + |T_{\mathcal{D}}|} \tag{6.65}$$

*is defined to be the **structural distance** between two poINS-F.*

This definition is totally equivalent to the structural distance between two poNS-F defined in Definition 6.10; thus, all the results we proved for the distance between poNS-F are still valid also for the structural distance between poINS-F. Let us see and example describing the encoding of two poINS-F, the assignment functions and the distance between the families.

Example 6.18. *Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ be two poINS-F such that $\mathcal{A}_{\langle I,<\rangle} = \{A_1, A_2, A_3, A_4, A_5, A_6, A_7\}$ and $\mathcal{D}_{\langle J,<\rangle} = \{D_1, D_2, D_3, D_4, D_5, D_6, D_7\}$. We can see a graphical representation of these families in Figure 6.19.*

Let $T_{\mathcal{A}} = \{t_1^{\mathcal{A}}, t_2^{\mathcal{A}}, t_3^{\mathcal{A}}, t_4^{\mathcal{A}}, t_5^{\mathcal{A}}, t_6^{\mathcal{A}}, t_7^{\mathcal{A}}\}$ be the encoding poset of $\langle T_{\mathcal{A}}, <\rangle$ such that $t_1^{\mathcal{A}} = \langle 1,0,360\rangle$, $t_2^{\mathcal{A}} = \langle 2,0,180\rangle$, $t_3^{\mathcal{A}} = \langle 2,180,360\rangle$, $t_4^{\mathcal{A}} = \langle 3,0,90\rangle$, $t_5^{\mathcal{A}} = \langle 3,90,180\rangle$, $t_6^{\mathcal{A}} = \langle 3,180,270\rangle$ and $t_7^{\mathcal{A}} = \langle 3,270,360\rangle$.

Let $T_{\mathcal{D}} = \{t_1^{\mathcal{D}}, t_2^{\mathcal{D}}, t_3^{\mathcal{D}}, t_4^{\mathcal{D}}, t_5^{\mathcal{D}}, t_6^{\mathcal{D}}, t_7^{\mathcal{D}}\}$ be the encoding poset of $\langle T_{\mathcal{D}}, <\rangle$ such that $t_1^{\mathcal{D}} = \langle 1,0,360\rangle$, $t_2^{\mathcal{D}} = \langle 2,0,180\rangle$, $t_3^{\mathcal{D}} = \langle 2,180,360\rangle$, $t_4^{\mathcal{D}} = \langle 3,0,180\rangle$, $t_5^{\mathcal{D}} = \langle 3,180,240\rangle$, $t_6^{\mathcal{D}} = \langle 3,240,300\rangle$ and $t_7^{\mathcal{D}} = \langle 3,300,360\rangle$.

Let $f : \langle T_{\mathcal{A}}, <\rangle \to \langle T_{\mathcal{D}}, <\rangle \cup \{t_0\}$ be the assignment function and d_f the binary assignment function; in Figure 6.19 we can see a graphical representation of this function. The binary assignment function assigns a one value (d_f) to the null assignment and a zero value to the elements in $T_{\mathcal{A}}$ which have a correspondent element in $\langle T_{\mathcal{D}}, <\rangle$.

Then:

$$d_S(\mathcal{A}, \mathcal{D}) = \frac{\sum_{t_i^{\mathcal{A}} \in T_{\mathcal{A}}} d_f(t_i^{\mathcal{A}}, f(t_i^{\mathcal{A}})) + |T_{\mathcal{D}} \setminus R(f)|}{|T_{\mathcal{A}}| + |T_{\mathcal{D}}|} = \frac{4+4}{14} = 0.5714$$

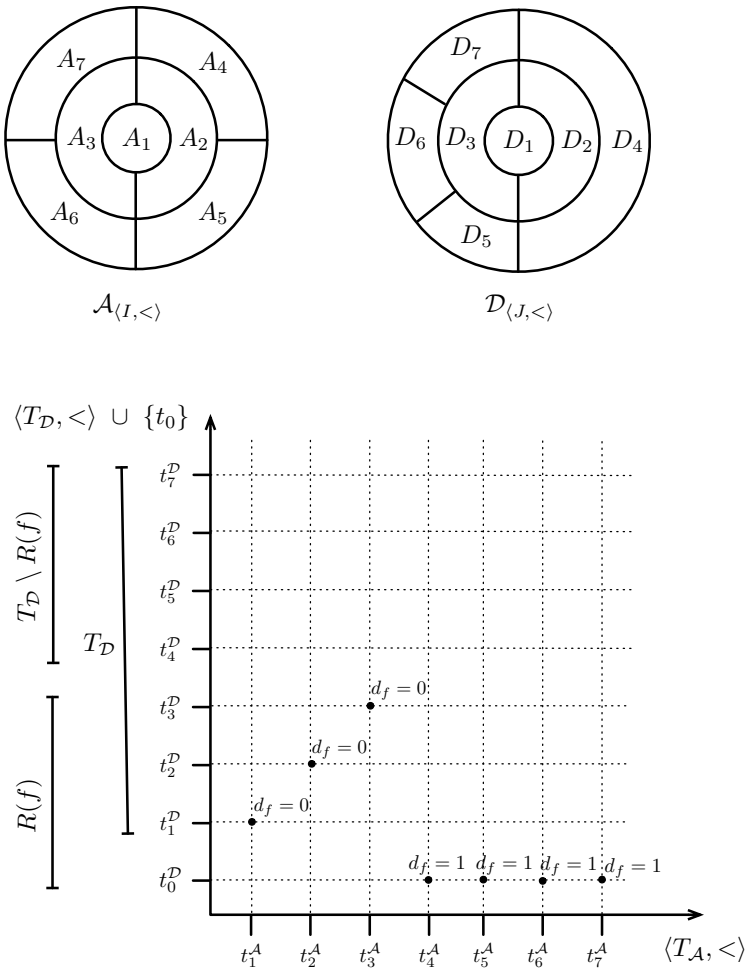


Figure 6.19: The DocBall representation of the two poINS-F described in Example 6.18 and the graphical representation of the assignment function used to compute the structural distance between these two families.

6.6.4 The NESTOR Distance

We have defined the graphical distance that is used to compute the distance between two sets – either in the NS-M or in the INS-M – and two other distance measures which are used to calculate the distance between two families. The first is the Jaccard's distance which is used to define a distance based on the elements belonging to the families and the second is the structural distance which defines the distance only on a structural basis. We may need to calculate a distance between two families of sets which takes into account both these measures; we define the so called "*NESTOR distance*" (d_N) as the parametrized linear combination of the Jaccard's distance and the structural distance. We define this distance for the NS-M and the INS-M; as we have seen the only difference between the two – from the distance point-of-view

– is the encoding of the sets which influences the structural distance.

Definition 6.12.
Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ be two partially ordered families of sets – i.e. poNS-M or poINS-M, $d_J(\mathcal{A},\mathcal{D})$ be the Jaccard's distance and $d_S(\mathcal{A},\mathcal{D})$ be the structural distance between them. Then, the NESTOR distance is defined to be:

$$d_N(\mathcal{A},\mathcal{D}) = \alpha\, d_J(\mathcal{A},\mathcal{D}) + (1-\alpha)\, d_S(\mathcal{A},\mathcal{D}) \tag{6.66}$$

where $0 \leq \alpha \leq 1$.

As we have seen in Section 4.1.6, the measure of distance is associated with a measure of similarity. From the NESTOR distance it is immediate to determine a **coefficient of similarity** between two families of sets.

Definition 6.13.
*Let $\mathcal{A}_{\langle I,<\rangle}$ and $\mathcal{D}_{\langle J,<\rangle}$ be two partially ordered families of sets – i.e. poNS-M or poINS-M, $d_N(\mathcal{A},\mathcal{D})$ be the NESTOR distance between them. Then, the **NESTOR coefficient of similarity** is:*

$$N(\mathcal{A},\mathcal{D}) = 1 - d_N(\mathcal{A},\mathcal{D}). \tag{6.67}$$

Now, we have to prove that the NESTOR distance is a proper metric for the set data models: NS-M and INS-M.

Lemma 6.25. *Let $\mathcal{A}_{\langle I,<\rangle}$, $\mathcal{D}_{\langle J,<\rangle}$ and $\mathcal{O}_{\langle K,<\rangle}$ be three partially ordered families of sets. Let $d_N(\mathcal{A},\mathcal{D})$, $d_N(\mathcal{A},\mathcal{O})$ and $d_N(\mathcal{D},\mathcal{O})$ be the NESTOR distances between the families. Then, d_N is a **proper metric**.*

Proof:
Since d_J and d_S are proper metrics and d_N is a parametrized linear combination of them, identity, non-negativity and symmetry are trivially true for d_N. The triangle inequality holds for d_J and d_S, then:

$d_N(\mathcal{A},\mathcal{O}) + d_N(\mathcal{D},\mathcal{O}) \geq d_N(\mathcal{A},\mathcal{D}) \Rightarrow \alpha\, d_J(\mathcal{A},\mathcal{O}) + (1-\alpha)\, d_S(\mathcal{A},\mathcal{O}) + \alpha\, d_J(\mathcal{D},\mathcal{O}) + (1-\alpha)\, d_S(\mathcal{D},\mathcal{O}) \geq \alpha\, d_J(\mathcal{A},\mathcal{D}) + (1-\alpha)\, d_S(\mathcal{A},\mathcal{D}) \Rightarrow \alpha\,(d_J(\mathcal{A},\mathcal{O}) + d_J(\mathcal{D},\mathcal{O})) + (1-\alpha)(d_S(\mathcal{A},\mathcal{O}) + d_S(\mathcal{D},\mathcal{O})) \geq \alpha\, d_J(\mathcal{A},\mathcal{D}) + (1-\alpha)(d_S(\mathcal{A},\mathcal{D}))$.

We know that: $d_J(\mathcal{A},\mathcal{O})+d_J(\mathcal{D},\mathcal{O}) \geq \alpha\, d_J(\mathcal{A},\mathcal{D}) \Rightarrow \alpha\,(d_J(\mathcal{A},\mathcal{O})+d_J(\mathcal{D},\mathcal{O})) \geq \alpha\, d_J(\mathcal{A},\mathcal{D})$, and that $d_S(\mathcal{A},\mathcal{O})+d_S(\mathcal{D},\mathcal{O}) \geq d_S(\mathcal{A},\mathcal{D}) \Rightarrow (1-\alpha)(d_S(\mathcal{A},\mathcal{O})+d_S(\mathcal{D},\mathcal{O})) \geq (1-\alpha)(d_S(\mathcal{A},\mathcal{D}))$.
□

Since d_N is a proper metric for the NS-M and the INS-M we can present these models as **metric spaces**.

Theorem 6.26. *Let $\mathcal{NSM} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ be a collection of poNS-F and d_N be the NESTOR distance. Then, $(\mathcal{NSM}, d_N)$ where $d_N : \mathcal{NSM} \times \mathcal{NSM} \to \mathbb{R}$, is a* ***metric space****.*

Proof:
In order to show that $(\mathcal{NSM}, d_N)$ is a metric space, it suffices to show that d_N is a proper metric for the domain of objects under consideration. The collection $\mathcal{NSM}$ is composed by poNS-F and Lemma 6.25 shows that d_N is a proper metric for partially ordered families of sets.□

Exactly in the same way we can define the INS-M as a metric space.

Theorem 6.27. *Let $\mathcal{INSM} = \{\mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n\}$ be a collection of poINS-F and d_N be the NESTOR distance. Then, $(\mathcal{INSM}, d_N)$ where $d_N : \mathcal{INSM} \times \mathcal{INSM} \to \mathbb{R}$, is a* ***metric space****.*

Proof:
In order to show that $(\mathcal{INSM}, d_N)$ is a metric space, it suffices to show that d_N is a proper metric for the domain of objects under consideration. The collection $\mathcal{INSM}$ is composed by poINS-F and Lemma 6.25 shows that d_N is a proper metric for partially ordered families of sets.□

Since the Jaccard's distance and the structural distance have been defined for the NS-M and the INS-M and they have been proved to be proper metrics, they can be adopted to define metric spaces based on them. We choice to adopt the NESTOR distance because it comprises both the Jaccard and the structural distances.

Chapter 7

The NESTOR Prototype and its Implementation in the SIAR System

In this chapter we present the NESTOR Prototype which describes a set-based application of the NESTOR Model. In Figure 7.1 we can see the main components composing the prototype and its implementation in the SIAR system.

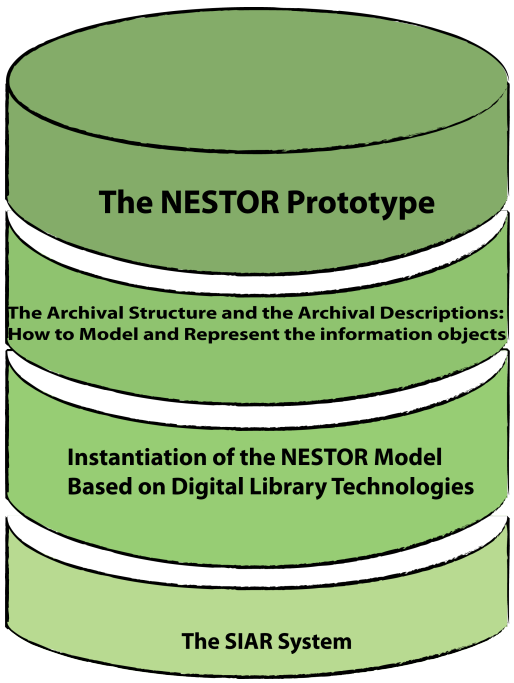


Figure 7.1: The main components of the NESTOR Prototype and its implementation in the SIAR System.

Initially we describe how an archive is modeled by means of the set data models presented in Chapter 6 and then, we present the requirements we have to fulfill in order to meet the

interoperability dimensions of Digital Libraries we described in Chapter 5.

Afterwards, we present two possible applications of the NESTOR Model. The first one is an *XML tree-based application* which relies on the standard encoding for archival descriptions: the Encoded Archival Description (EAD). The second is a *set-based application* which relies on the joint use of OAI-PMH and Dublin Core. We present advantages and disadvantages of these applications focusing on the requirements we have to fulfill.

These applications allow us to point out the relationships with the Digital Libraries technologies – see Section 5.3 – emphasizing how the NESTOR Framework exploits existing and widely-adopted standard technologies. This aspect highlights the characteristics of flexibility and adaptability of the framework in concrete application environments.

Lastly, we present an archival information system implementing the NESTOR Prototype: the *SIAR (Sistema Informativo Archivistico Regionale)* [Agosti et al., 2008, 2007b,c; Ferro and Silvello, 2008a, 2009a]. The SIAR is an information system designed and developed in the context of the Italian Veneto Region that requires the creation, management, access and exchange of archival metadata in a distributed environment.

7.1 How to Model an Archive

In order to model an archive we have to take into account its main characteristics which are: its structure and the objects it manages and preserves. As we have seen in Chapter 5, an archive is a complex organization composed by several parts. The foremost component regards the descriptive part of an archive which is conceptually modeled by the ISAD(G) standard; this standard defines the hierarchical organization of archival descriptions and how to model the relationships between them.

We can point out two main aspects that we have to consider when modeling an archive: *hierarchy* and *context*. The first aspect means that we have to be able to represent and maintain the hierarchical structure of an archive and its descriptions; the second aspect means that we have to retain the relationships between the archival descriptions and to exploit them to reconstruct the context of a document in relationships with its creation and preservation environment. In order to express hierarchy and context we need to dispose of a model which allows us to represent the structure of an archive. Furthermore, we need to represent also the content of an archive which is described and managed by means of archival descriptions – that in a digital environment are represented by archival metadata.

As we have seen in Chapter 5, the structure of an archive is usually represented by means of a tree where each node represents an archival division – e.g. a fonds, a sub-fonds, a series[1] – and the edges connecting the nodes represent the relationships between the divisions. These relationships constitute the hierarchical structure of the archive. Let us consider the next example where a sample archive is modeled by means of a tree.

[1] Please note that the terms *fonds* and *sub-fonds* are used both for the singular and plural form of the noun – see Section 5.2. The term series is used both for indicating one series and many distinct series.

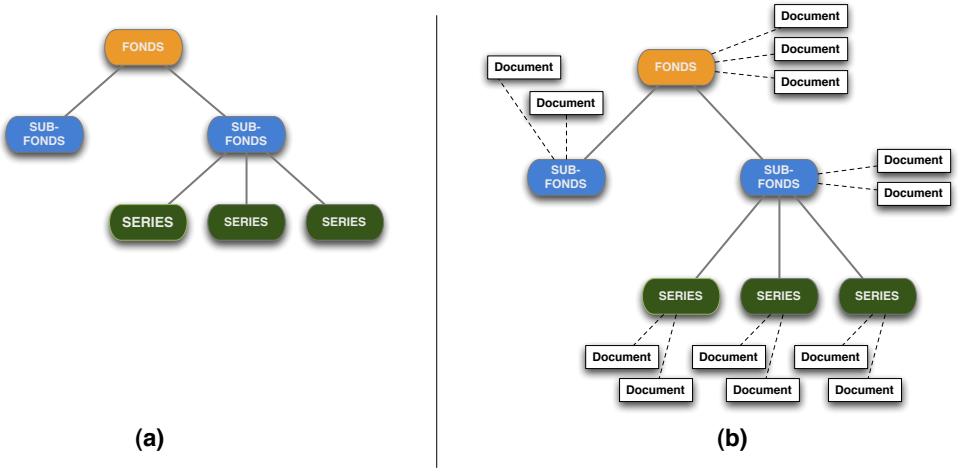


Figure 7.2: (a) An archive represented by means of a tree. (b) An archive and its documents represented by a tree.

Example 7.1. *Let us consider an archive composed by six divisions: a fonds, two sub-fonds and three series. The fonds is divided into two sub-fonds such that the first has no sub-divisions and the second contains the three series. The fonds contains three documents, both the sub-fonds contain two documents and each series contains two documents. Every document is described by an archival description.*

Following the ISAD(G) principles we have to represent this archive as a level hierarchy – please see Chapter 2 – where the fonds belongs to the higher level, the sub-fonds to the next lower level and so on for the series. In the archival practice this archive is represented by means of a tree, as we can see in Figure 7.2a.

As we can see in Figure 7.2a, the tree does not allow us to represent the documents belonging to each division. If we want to represent the documents we need to extend the tree model with some additional nodes attached to each node representing an archival division. These nodes have a different meaning respect to the nodes representing a division of an archive; they are contained by the node representing an archival division – e.g. the node representing the fonds contains three documents and in turn it is divided into two sub-fonds – but they have to be distinguishable from the nodes representing other archival subdivisions. With the representation in Figure 7.2b, the sub-fonds and the documents contained by the fonds are at the same hierarchical level and they seem to have the same meaning. In order to enhance the differences between these nodes, we drew dotted edges for the nodes representing a document and not an archival division.

From this example we can see that the tree model is adequate to represent the structure of an archive because it properly represents the hierarchical relationships between the archival divisions; on the other hand, in a tree it is not straightforward to represent the documents belonging to each archival division. We can say that the tree can represent the structural aspects of an archive but it needs to be somehow extended in order to represent also the content – i.e.

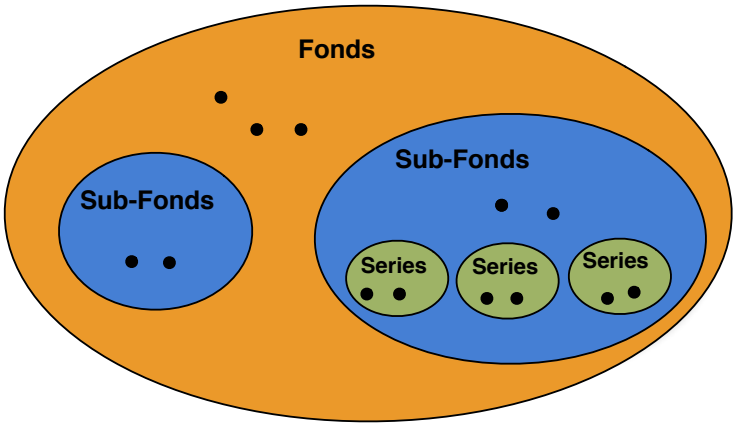


Figure 7.3: An archive modeled by means of the NS-M.

the archival resources.

One of the main features of the NESTOR Model is the possibility to express both the hierarchical structure by means of the nested sets and the content by means of the elements belonging to the sets. By means of the NESTOR Model, the archival divisions are represented as nested sets and the hierarchical relationships are retained by their inclusion order. On the other hand, the archival resources are represented as elements belonging to the sets. Let us see an example where the same archive considered in the Example 7.1 is modeled by means of the NS-M.

Example 7.2. *Let us consider the archive presented in the Example 7.1 and in Figure 7.3 we can see that archive represented by a NS-F where each set corresponds to an archival division. From this representation we can see that the division "fonds" is divided into two sub-divisions represented by the two subsets on the set "fonds", and so on and so forth for the series.*

At the same time, the documents belonging to each division are represented by means of elements belonging to the sets with a clear distinction between the structural and content components.

The NS-M allows us to straightforwardly represent an archive; from the Theorem 6.8 presented in Chapter 6, we know that a tree can be mapped into a NS-F and thus we know that its expressive power is preserved by the NS-M. In this case we can see that the NS-M allows us to define a further level of expressiveness respect to the tree.

In the NESTOR Model we can go from the NS-M to the INS-M and the two models have the same expressive power; indeed, we can model an archive by means of the INS-M as well as we have done with the NS-M. Figure 7.4 shows an archive modeled by an INS-F represented throughout a DocBall. Each circular section of the DocBall represents a set and the archival documents are represented by the elements contained in each section.

The set data models are well-suited for the archival practice; indeed, the idea of "set" shapes

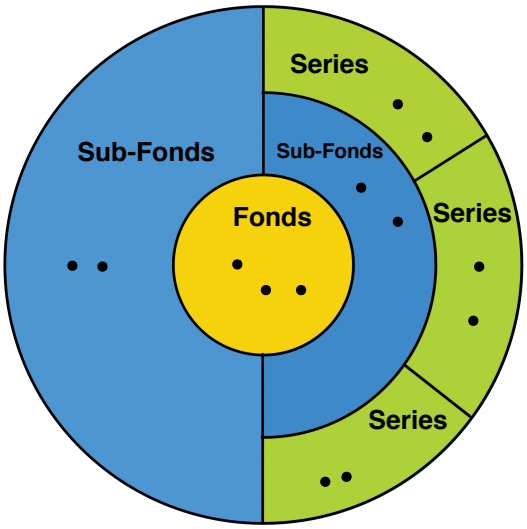


Figure 7.4: An archive modeled by means of the INS-M.

the concept of archival division which is a "container" comprising distinct elements that have some properties in common. If we think to the Chinese boxes metaphor (see Section 2.1.1), a hierarchy is composed by a sequence of boxes contained one into the others; if we look to an archive from the physical point-of-view, we can see that it resembles the Chinese boxes structure: There are boxes, folders, sheets, etc. one contained inside the others. The concept of nested sets is closer to this view of the reality than the concept of tree and the archivists have the possibility to choose the set data model – i.e. NS-M or INS-M – that best fits their way of describing the archives.

7.2 Analysis of the Requirements

We have seen how archives can be modeled by means of the NESTOR Model highlighting some advantages of this approach. The goal of the NESTOR Prototype is to offer a concrete instantiation of the data models described so far; such an instantiation has to meet several requirements which are strictly related to the world of Digital Libraries [Agosti and Ferro, 2010]. The list of requirements that has to be fulfilled is derived from an analysis of the dimensions of interoperability of Digital Libraries and from the user studies we have conducted with archivists in the context of the design and development of the NESTOR Prototype.

7.2.1 Information Objects

We have to settle the objects described by the model and what we want to do with them. We have to take into account the nature of an archive and the objects corresponding to the archival resources in the digital environment. First-of-all, in a digital environment from the archival point-of-view, we have to deal with two kinds of digital objects: *descriptive metadata* and *full*

content digital objects. Descriptive metadata are the foremost resources managed by digital archives because often the archival resources are not available in digital form, but they are described and represented by metadata; sometimes archival resources are metadata themselves. In recent times the availability of full content archival digital objects has been growing and thus they have to be modeled as well as the descriptive metadata.

In the context under examination we assume the set of digital resources (we write only "resources" when there is no need to specify their digital nature) as conceptually divided into metadata and digital objects. A *metadata* is a resource that describes an object that may or may not have a digital form; we indicate with $M = \{m_1, \ldots, m_k, \ldots, m_t\}$ the set of metadata and with $m_k \in M$ a metadata m_k belonging to M. If an object described by a metadata has a digital form, we call it a *digital object* that can have different representations such as a text file, an audio or video file or a photograph; it is not mandatory for a digital object to be described by a metadata. We indicate with $DO = \{do_1, \ldots, do_k, \ldots, do_s\}$ the set of digital objects and with $R = M \cup DO = \{r_1, \ldots, r_k, \ldots, r_{t+s}\}$ the set of resources given by the union of the metadata set and the digital object set[2].

For what is concerned with metadata, in the archival context we have to be able to handle different metadata formats in order to deal with different informative needs. Indeed, different archives may use different metadata formats to describe their resources or within the same archive there may be the necessity to describe different resources by means of different metadata formats [Prom et al., 2007].

An important requirement is the possibility to update the archival metadata, change the format of the metadata or add new metadata formats without interfering with the archival structure. Vice versa, it is very important to be able to change the structure of an archive – e.g. add an archival division, join together or divide two or more archival divisions – without requiring any changes in the metadata. The *clear distinction between content and structure is a key requirement* that the NESTOR Prototype has to meet.

When we consider also full-content digital objects, we have to be able to relate every digital object with the archival division at which it belongs as well as it is mandatory to do with descriptive metadata. Furthermore, it is necessary to dispose of a mechanism to define and exploit the relationship between a digital object and the metadata describing it. We have to consider the necessity to deal with aggregations of resources which are composed both by metadata and digital objects. We can see that in this context we can have a descriptive metadata related to a non-digital archival resource, a descriptive metadata related to an archival digital object, and a descriptive metadata describing an aggregation of resources.

7.2.2 Functional Perspective

From the functional perspective point-of-view we take into account several fundamental aspects that the NESTOR Prototype has to fulfill: access, exchange, manipulation, and querying of the archival resources – i.e. descriptive metadata and full-content digital objects. First-of-all

[2]Please note that the intersection between the set M of metadata and the set DO of digital object is empty.

we consider these aspects from the metadata point-of-view and then, we extend the requirements to enclose also the digital objects.

Access: The archival descriptions have to be accessible from multiple entry points and at the same time they have to disclose their relationships allowing the user to consult contextual information.

Exchange: The archival descriptions have to be shareable in a distributed environment with a variable granularity and have to provide a mechanism for reconstructing the hierarchy when necessary.

Manipulate and Query: The users must have a means at their disposal for manipulating both the archival structure and the archival descriptions, and for defining and performing queries.

The access requirement states that we have to be able to consult an archive starting from the required description without having to navigate the whole archival hierarchy from a unique entry point to find the information of interest. At the same time from each description we have to be able to reconstruct its relationships with the other elements of the archive – i.e. preserving and exploiting the archival bonds.

The exchange requirement states that we have to be able to exchange archival descriptions with different degrees of coarseness and belonging to whatever level of the archival hierarchy without having to exchange the whole archive. Furthermore, a mechanism needs to be available for reconstructing the archival relationships of an exchanged description whenever it is necessary. In the current state of development of Digital Libraries an important technological requirement is compliance with OAI-PMH. Through the fulfillment of this requirement we can address the "*functional perspective*" and the "*interoperability technology*" aspects of interoperability.

When we consider also full-content digital objects, we have to add a further aspect to this vision; indeed, when we consider the set R of resources we have to deal with resources that can be seen as "atoms" or "aggregations". We have to be able to access and exchange with variable granularity both atoms and aggregations of archival resources retaining both the relationships with the archival structure and between the objects. In this context, we rely on the concept of *handle*, which is defined as a unique identifier of both metadata and digital objects [Agosti and Ferro, 2007; Ferro and Silvello, 2010]. We assume that every archival resource is identified by a unique handle.

Definition 7.1.
Let H be a set of handles and R a set of resources such that $|H| = |R|$, where $h \in H$ is a generic handle. We define a bijective function $\eta : H \rightarrow R$, which maps a handle to the resource identified by it: $\forall r_i \in R, \exists! h_i \in H \mid \eta(h_i) = r_i \Rightarrow \eta^{-1}(r_i) = h_i$.

In order to manage, access and exchange archival resources and aggregations we have to be able to:

- assign a handle to each component we think should be accessed and exchanged in an independent way;
- retain the context and the structure of resources;
- define a machine-readable and standard mechanism for defining and treating aggregations of resources.

These aspects are central also from the user perspective which is focused on the manipulation and querying of the archives. When we need to manipulate and query an archive we have to be able to express constraints on the structure and on the content of an archive; to do so we need to have a well-defined mechanism that allows us to express our needs in a standard way. The manipulation operations that must be available to the users are the possibility to update, insert and delete an archival resource, an archival division or a whole part of an archive. The mechanism by means of which a user queries an archive must be independent from the particular technology of choice – e.g. the metadata format encoding the archival descriptions.

7.2.3 Multilingualism

Language techniques allowing cross-language access to the resources have to be straightforwardly applicable to the archival descriptions.

There is a huge need to provide cross-language access to information; this is due to the diversity and multilinguistic environment in which Digital Libraries are operating – e.g. in the European Union there are 23 official languages spoken in 27 member states[3]. Cross-language access to information leads to problems of both semantic and syntactic interoperability [Levergood et al., 2008]. It is necessary to address two "metadata-related challanges" that usually are faced by involving the specification of the language of the metadata fields [Levergood et al., 2008]: false friends and term ambiguity. Another important issue is "name resolution" which regards the necessity to disambiguate between words that are proper names that do not require a translation or nouns that has to be translated for multilingual purposes. For instance, the term "Bush" can be seen as the surname of a President of the United States or as a noun indicating a shrub. On the other hand, we may have the necessity to translate some proper names; for instance, the proper name"Kepler" has to be translated as "Keplero" in Italian.

To address these issues we can point out three main solutions usually employed in the context of Digital Libraries:

1. **Translation**: A query formulated in the user language is automatically translated in the other languages supported and then submitted to the system. This solution is not free from the false friends, name resolution and term ambiguity issues.

[3] `http://ec.europa.eu/education/languages/languages-of-europe/`

2. **Enrichment of Metadata**: This is understood as making the intended meaning of information resources explicit and machine-processable, thus allowing machines and humans to better identify and access the resources. The language would be thus provided in the metadata itself.

3. **Association to a Class**: This is the association of terms to a fairly broad class in a library classification system such as the Dewey Decimal Classification. This is a common solution for the term ambiguity problem. More advanced language techniques such as semantic annotation and tagging may be also taken into account and related to this solution.

The specification of the language of metadata field permits us to fully exploit metadata for cross-language purposes. If metadata do not come with or cannot be enriched with the language of the field, it is useful to rely on the association to a class technique. This technique relies on the use of the subject field of metadata; it is not always possible to determining the subject of a metadata or of a term. This is particularly true for archival metadata where determine the subject can be very difficult. In order to apply this technique to archival descriptions we have to be able to access and manipulate each description as a single independent entity; while this requirement is quite straightforward for libraries, it is not in the archival environment where all the descriptions are related to each other in a hierarchical structure.

7.3 An Application based on an XML Tree

EAD is the standard to encode archival descriptions and it is widely adopted to represent and manage the archives in a digital environment. As we have seen in Chapter 5, it is quite straightforward to instantiate an archive modeled by means of a tree into an EAD file. But, if we model an archive by means of the NESTOR Model the possibility to instantiate it by means of EAD is not precluded either.

In Figure 7.5 we can see how the archive of Example 7.1 modeled by means of the NS-F represented in Figure 7.3 can be mapped into an EAD file. The order inclusion between the sets defining the hierarchical relationships between the archival divisions is retained in the EAD by means of nested tags in the XML file. The elements representing the archival descriptions are encoded by a sub portion of XML nested inside each tag representing the corresponding archival division. Basically, this instantiation of the NESTOR Model represents the state-of-the-art in the representation and managing of archives in a digital environment. Figure 7.5 represents the mapping of a NS-F into a tree encoded by means of a unique XML file.

The main feature of this instantiation of the model is that both the structural and the content elements are represented by means of XML elements (i.e. tags). The EAD metadata allows us to encode the description part defined by means of the data models and thus it is a proper means for representing an archive. On the other hand, EAD imposes a unique metadata format to encode the archival descriptions – i.e. the EAD itself. In this context we cannot change the

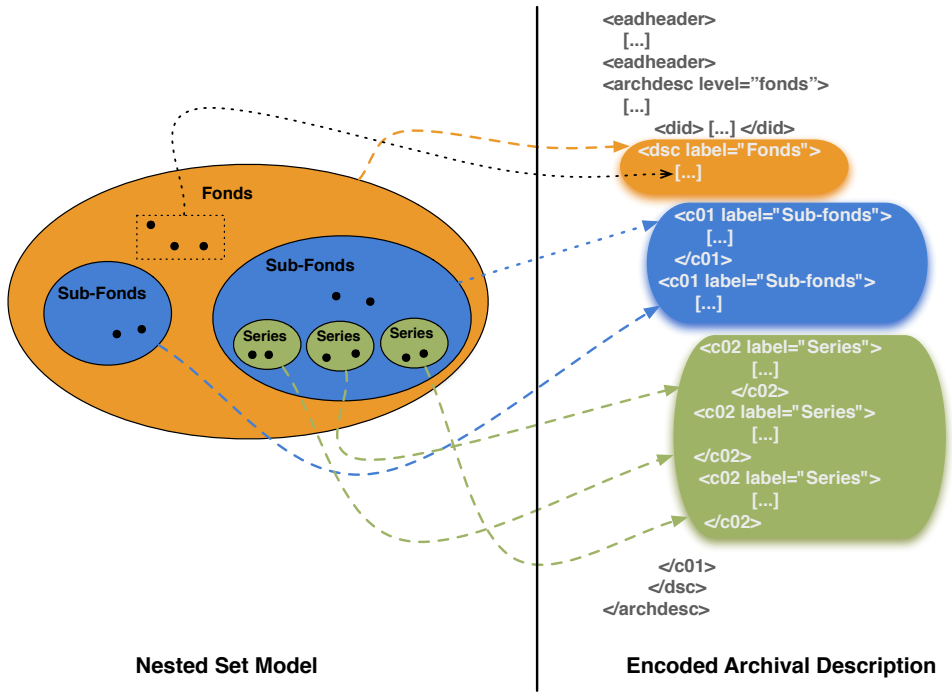


Figure 7.5: A sample archive represented throughout the NSM and mapped into an EAD file.

metadata format or add a further descriptive metadata; an archival resources can be described only by means of EAD and every description is embedded in the archival structure. This means that *content and structure are interlinked in the same XML file* and they cannot be handled separately. In this context it is impossible to update, insert or delete a descriptive metadata without interfering also with the structure of the archive. Thus, the first requirement regarding archival metadata is not respected.

This problem is emphasized when we consider the "functional perspective". The access requirement states that we have to be able to access the archival descriptions – i.e. descriptive metadata – at different degrees of coarseness. EAD is encoded as a unique XML file which mixes structural and content information while the entry point to access the information is the root of the XML file. From the root we have to navigate the hierarchy to access the information of interest. In order to overcome this issue we can define some superstructures to the EAD; for instance, we can settle some predefined entry points by the use of XPointers[4] pointing to specific elements of the XML or by using predefined paths driving the user through the hierarchical structure. These solutions are palliatives because they are not general solutions but they can only adequately match a specific reality with limited needs. Furthermore, for each instantiation of the EAD, we have to know in advance how the XML elements are used; this is not a problem in general because we can make use of the EAD schema, but to do so each instantiation of EAD has to meet stringent best practice guidelines [Prom et al., 2007] otherwise the use of tags may be inconsistent, leading to wrong interpretations of the information as has

[4] http://www.w3.org/TR/xptr-framework/

happened in practice [Kim, 2004].

The exchange requirement states that we have to be able to exchange archival descriptions in a distributed environment. The same issues affecting EAD for the access requirement can be found here for metadata exchange; indeed, the encoding of all the archival descriptions as a unique XML file forces us to exchange the archive as a whole. We cannot share a specific piece of information – e.g. the descriptions of the documents belonging to a specific "series" – without extracting it from the XML file and losing in this way the structural information retained thanks to the nested tags in the XML itself.

This issue is emphasized when we take into account the *digital objects*. As we have seen an EAD file encode both the structure of an archive and the descriptive metadata describing the archival resources; if some descriptive metadata describe digital objects belonging to the archive itself, we need to handle not only the metadata but also the relationships with the digital objects.

Let us consider the archive we described in the Example 7.1 modeled by means of a NS-F. Let us call A_1 the set representing the archival division "fonds", A_2 the set representing the first "sub-fonds", and A_3 the set representing the first "series".

Then, we assume that A_1 contains a digital object such that $A_1 = \{do_1\}$, A_2 contains $k \in \mathbb{N}$ digital objects such that $A_2 = \{do_2, \dots do_{k+1}\}$ and A_3 other k digital objects such that $A_3 = \{do_{k+2}, \dots do_{2k+2}\}$.

Every digital object can be associated with a metadata describing it or at least with the metadata describing the archival division at which it belongs. In context of the EAD instantiation a single EAD metadata describes not only the digital objects but also the structure of the archive retaining the relationships between the digital objects themselves; thus it is to relate each digital object with the component of EAD describing it. Only maintaining all these relationships we can reconstruct the full informative power of an archival resource. In Figure 7.6 we represent the sample archive we are taking into account; we highlight the three archival divisions under examination associated with the digital objects. We can see that the component "fonds" describes one digital object, instead, for instance, "sub-fonds" describes a bunch of digital objects.

This problem has also been recently tackled by two projects funded by the European Commission: Europeana[5] and APEnet[6]. In the context of these projects the solution showed in Figure 7.7 has been proposed [Sugimoto and van Dongen, 2009]; [Sugimoto and van Dongen, 2009] takes into account two scenarios: in the first one a component of EAD describes only one digital object, in the second one a single component describes a bunch of digital objects. In the *first scenario* a *Uniform Resource Locator (URL)* to the digital object do_1 is embedded into an element of the EAD component describing it. This approach in not flexible because if the URL of do_1 changes the content of the metadata must also be updated; furthermore, the relationships between digital objects cannot be automatically determined, instead they must be inferred by browsing the EAD metadata structure. The *second scenario* also has to address the problem

[5] `http://www.Europeana.eu/portal/`

[6] `http://apenet.net/`

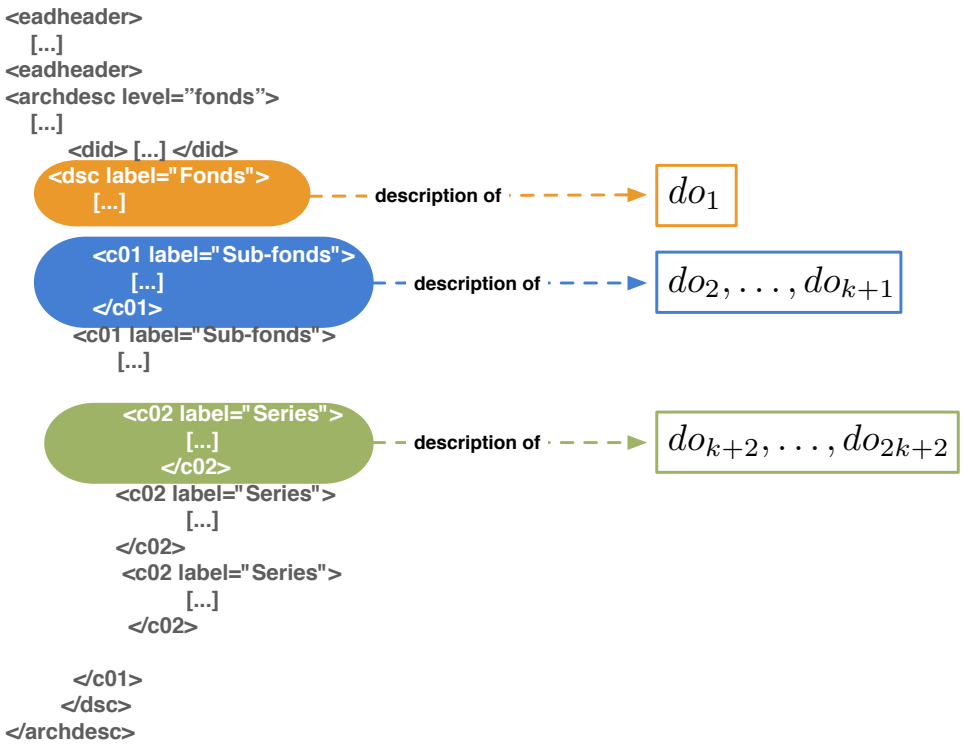


Figure 7.6: An EAD file associated with a bunch of digital objects.

that only one URL can be specified for each descriptive component of EAD; so, a *Metadata Encoding and Transmission Standard (METS)*[7] metadata format is used to connect the EAD descriptive component to the digital objects [Sugimoto and van Dongen, 2009]. Basically, the METS metadata acts as an in-between component relating each digital object with the EAD component describing it.

With this solution the relationships between the metadata and the digital objects as well as the relationships between the digital objects are defined in a cumbersome way that can lead to inconsistencies. For instance, in order to reconstruct the context of a digital object exchanged in a distributed environment a potential user must follow the URL from the digital object to the METS file, activate the URI to the EAD metadata and then browse the EAD tree to find out the component describing the digital object. The components of the metadata are not directly and independently identifiable, accessible and exchangeable and the digital objects are not related one to another in a way that permits us to automatically infer the relationships between themselves and between them and their descriptions.

Lastly, we analyze the possibility of using language techniques with the archival metadata. The first language technique (e.g. "translation") can be straightforwardly applied; furthermore, when we consider the translation of an EAD file we have the advantage to dispose of a big file with many contextual information which can be used to disambiguate the terms. On the other

[7] `http://www.loc.gov/standards/mets/`

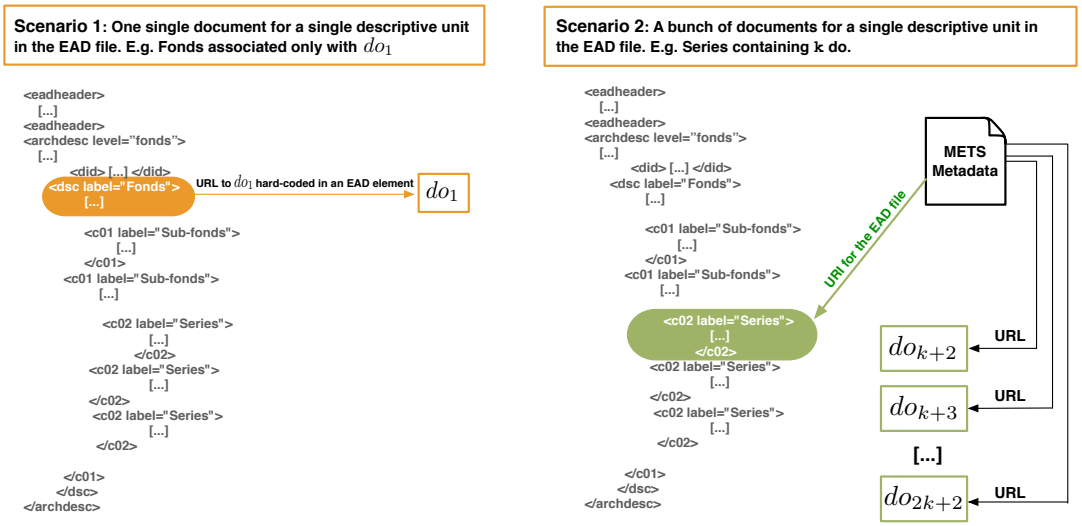


Figure 7.7: How to link the EAD file with the described digital objects.

hand, the "enrichment of metadata" technique requires the metadata to be machine-readable in order to be automatically processed and enriched. The very flexibility of EAD leading to a not always consistent use of structure and content elements precludes the possibility of adopting this technique with many EAD files. Finally, we know that a single EAD metadata is used to describe an entire archive, thus in a unique metadata we can find very different subjects. With this organization it is very difficult to disambiguate the terms or to identify the subject of metadata; with the EAD metadata the "association to a class" technique is essentially unworkable.

Another important point is the relationships with the standard Digital Library technologies: OAI-PMH, the Dublin Core Framework and OAI-ORE. As we have seen, OAI-PMH can be used only to exchange the whole archive as a monolithic unit and thus, many of the useful functionalities of the protocol cannot be straightforwardly exploited. In order to overcome this problem different solutions have been pointed out proposing different mappings of EAD into a collection of Dublin Core metadata that can be exchanged and accessed with a variable granularity [Bountouri and Manolis, 2009; Prom, 2003; Prom and Habing, 2002]. The main problem of these solutions is that the Dublin Core metadata cannot retain the archival structure by themselves but they have to be related by means of several links to the EAD structure and thus, they are not independent from the original EAD file. The same problem arises when we want to use EAD with OAI-ORE, thus preventing a full adoption of this technology.

As we have seen several requirements are not meet by the instantiation of the NESTOR Model by means of EAD; on the other hand, we can overcome some issues by pointing out some ad-hoc solutions that have to be defined on a case-by-case basis.

7.4 A Set-Based Application

In this section we describe an instantiation of the NESTOR Model based on the conjunction between standard Digital Libraries technologies: OAI-PMH and the Dublin Core. This application allows us to overcome the issues we presented in the previous section and to present a general solution to deal with hierarchies in the context of Digital Libraries. Indeed, we see that by exploiting the formal basis we defined and the very native features of existing Digital Library building blocks, we can enhance the flexibility, adaptability and scalability of the set data models pointing out general methodologies to meet the presented requirements.

The first requirement that a set-based application has to satisfy is the capacity to represent the archival structure and the archival resources. Firstly, we present the application considering only archival metadata and showing how it meets the requirements exploiting standard technologies such as OAI-PMH and Dublin Core [Ferro and Silvello, 2008b, 2009b]; afterwards, we describe how this very instantiation of the model can be adopted to deal also with digital objects exploiting another Digital Library standard technology which is OAI-ORE.

7.4.1 A Flexible Representation of Archival Descriptions

In OAI-PMH it is possible to define an OAI-Set organization based on the NS-M or INS-M. This means that we can treat the OAI-Sets as a Nested Set Family (NS-F) or as an Inverse Nested Set Family (INS-F). The inclusion order between the OAI-Sets is given by their identifiers which are `<setspec>` values. In the following we describe how it is possible to create a Nested Set family of OAI-Sets and afterward how the same thing can be done with an Inverse Nested Set family. In order to have a detailed description of the OAI-PMH functioning please refer to Section 5.3.

Let $\mathcal{F}_I$ be a NS-F where I be the index set composed by the `<setspec>` values such that $i \in I = \left\{s_0 : s_1 : \ldots : s_j\right\}$ means that $\exists\, F_j \in \mathcal{F}_I \mid F_j \subset \ldots \subset F_1 \subset F_0$. Every $F_j \in \mathcal{F}_I$ is an OAI-Set uniquely identified by a `<setspec>` value in I. The `<setspec>` values for the $F_k \in \mathcal{F}_I$ are settled in such a way to maintain the inclusion order between the sets. If an F_k has no superset its `setspec` value is composed only by a single value (`<setspec>`s_k`</setspec>`). Instead if a set F_h has supersets, e.g. F_a and F_b where $F_b \subset F_a$, its `setspec` value must be the combination of the name of its supersets and itself separated by the colon [:] (e.g. `<setspec>`$s_a : s_b : s_h$`</setspec>`). Furthermore, let $OAI = \{oai_0, \ldots, oai_n\}$ be a set of OAI records, then each $oai_i \in F_j$ must contain the setspec of F_j in its header.

Let us consider the sample archive represented by the NS-F in Figure 7.3. As we can see in Figure 7.8, each set composing this nested set structure is mapped into an OAI-Set with a proper `setSpec`; the set called "fonds" is mapped into an OAI-Set with < `setspec` > `0001` < /`setspec` >. This set has two subsets that are mapped into two OAI-Sets: < `setspec` > `0001` : `0002` < /`setspec` > and < `setspec` > `0001` : `0003` < /`setspec` > and so on for the other sets. We can see that the hierarchical relationships and thus the inclusion order between the sets is maintained by the identifiers of the OAI-Sets which are defined as materialized

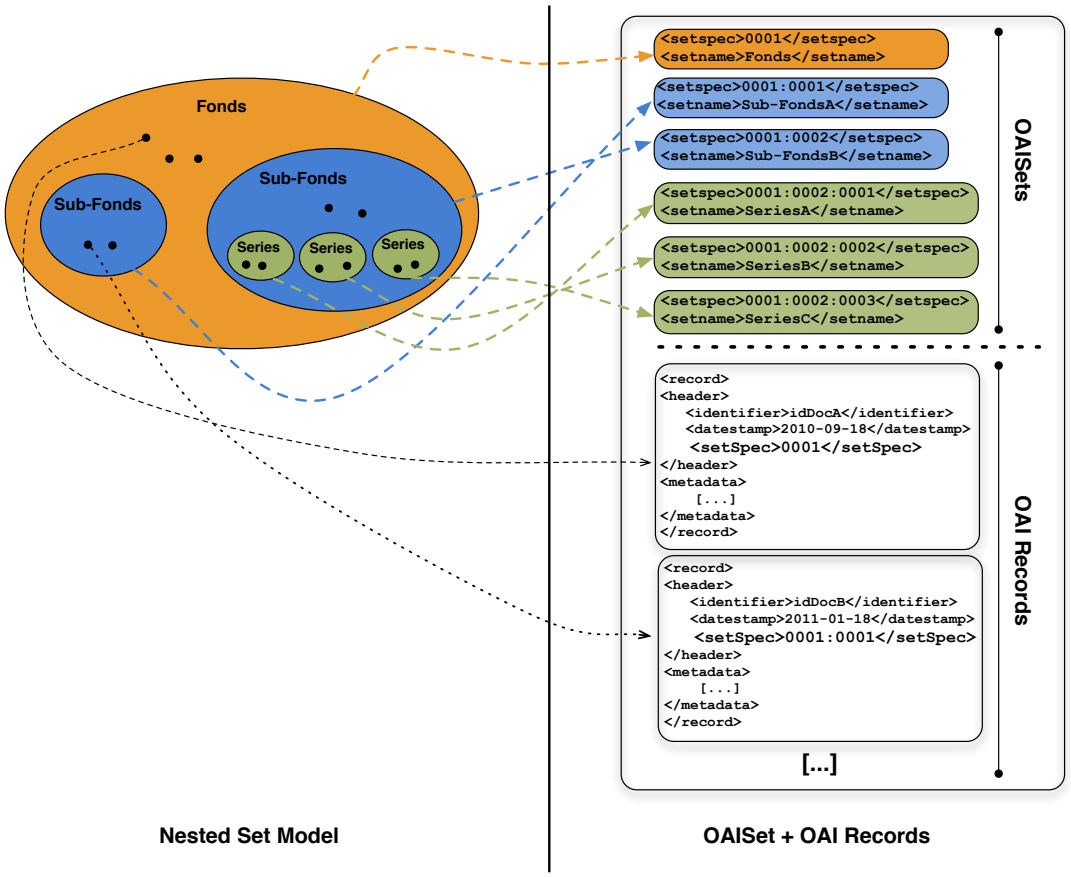


Figure 7.8: A sample archive represented throughout the NS-M and mapped into OAI-Sets and OAI Records.

paths from the root to the identified set. Each single archival description is mapped into a metadata belonging to an OAI-Set; the membership information is added to the header of these metadata that are seen as OAI-records. In this way each archival description can be encoded by a single metadata without any constraints on its format; indeed, an OAI-Set can contain different kinds of metadata formats. With this model we do not impose any conditions on the archival descriptions, thus allowing the possibility of changing the metadata, updating the information or adding a new metadata format without affecting the structure of the archive and without changing the data model. We choose the Dublin Core as minimum metadata requirement and this choice is lead by widespread use of this metadata in Digital Libraries and the possibility of defining Dublin Core *Application Profiles* which allow us to make it domain-specific; indeed, Dublin Core Application Profiles allow the definition of Dublin Core metadata formats well-suited for the reality we intend to represent.

An important aspect that has to be highlighted is that this instantiation maintains also the horizontal dimension of the archival hierarchy – i.e. the order between the subsets of a set. Indeed, Knuth stated in [Knuth, 1997]: “The very nature of computer representation defines an implicit ordering for any tree” – please refer to Section 2.2.2 – and in the same way the very

nature of computer representation defines an implicit ordering between the OAI-Sets. In Figure 7.8 we can see that we can talk of the first sub-fonds of the fonds (we named it Sub-fondsA) or of the second series a sub-fonds. This is possible because the OAI `setspecs` define the inclusion order between the OAI-Sets but also a partial order between the OAI-Sets which are common subsets of another OAI-Set. Thus, the NS-F $\mathcal{F}_I$ is actually a partially-ordered NS-F (poNS-F) because the index set of `setspecs` is a partially ordered set: $\langle I, < \rangle$.

In the same way we can apply the INS-M to OAI-PMH; Let $\mathcal{G}_J$ be an Inverse Nested Set family where J be the set of the <setspec> values such that $j \in J = \{s_0 : s_1 : \ldots : s_k\}$ means that $\exists\ G_k \in \mathcal{G}_J = G_k \subset \ldots \subset G_1 \subset G_0$. In $\mathcal{G}_J$ differently that in $\mathcal{F}_I$ the following case may happen: Let $\{G_i, G_k, G_w\} \in \mathcal{G}_J$ then it is possible that $G_w \subset G_i$ and $G_w \subset G_k$ but either $G_i \not\subseteq G_k$ and $G_k \not\subseteq G_i$. If we consider $\mathcal{G}_J$ composed only of G_i, G_k and G_w, the identifier of G_i is <setspec>s_i</setspec> and the identifier of G_k is <setspec>s_k</setspec>. Instead, the identifier of G_w must be <setspec>$s_i : s_w$</setspec> and <setspec>$s_k : s_w$</setspec> at the same time; this means that in $\mathcal{G}_J$ there are two distinct OAI-Sets, one identified by <setspec>$s_i : s_w$</setspec> and the other identified by <setspec>$s_k : s_w$</setspec>. This is due to the fact that the intersection between OAI-Sets in OAI-PMH is not defined set-theoretically [Van de Sompel et al., 2003]; indeed, the only way to get an intersection of two OAI-Sets is enumerating the records. This means that we can know if an OAI record belongs to two or more sets just by seeing whether there are two or more <setspec> entries in the header of the record. In this case the records belonging to G_w will contain two <setspec> entries in their header: <setspec>$s_i : s_w$</setspec> and <setspec>$s_k : s_w$</setspec>; note that only the <setspec> value is duplicated and not the records themselves.

Let us consider the sample archive represented by the NS-F in Figure 7.4. In Figure 7.9 we can see how the INS-F is mapped into a collection of OAI-Sets and OAI Records. We obtain four sets from the common subset – i.e. the fonds of the sample archive – with four different identifiers: “0004:0001”, “0001:0001:0001”, “0002:0001:0001” and “0003:0001:0001”. In the same way are defined the sets mapped from the children of the root. The sets related to the series are identified by “0001”, “0002” and “0003”. We can see that the OAI Records belonging to the “fonds” have four `setspecs` in the header because the fonds in the INS-M representation is the common subset of four other sets and thus, it has four different associated OAI-Sets.

These instantiations of the set data models have two main main relevant features which are also important aspects defining the *flexibility* and *adaptability* of the NESTOR Model: *they clearly divide the structural elements* (i.e. the sets) *from the content elements* (i.e. the archival descriptions) and *they do not bind the archival descriptions to a unique, fixed and predefined metadata format*. These differences have a major impact in the fulfillment of the other presented requirements.

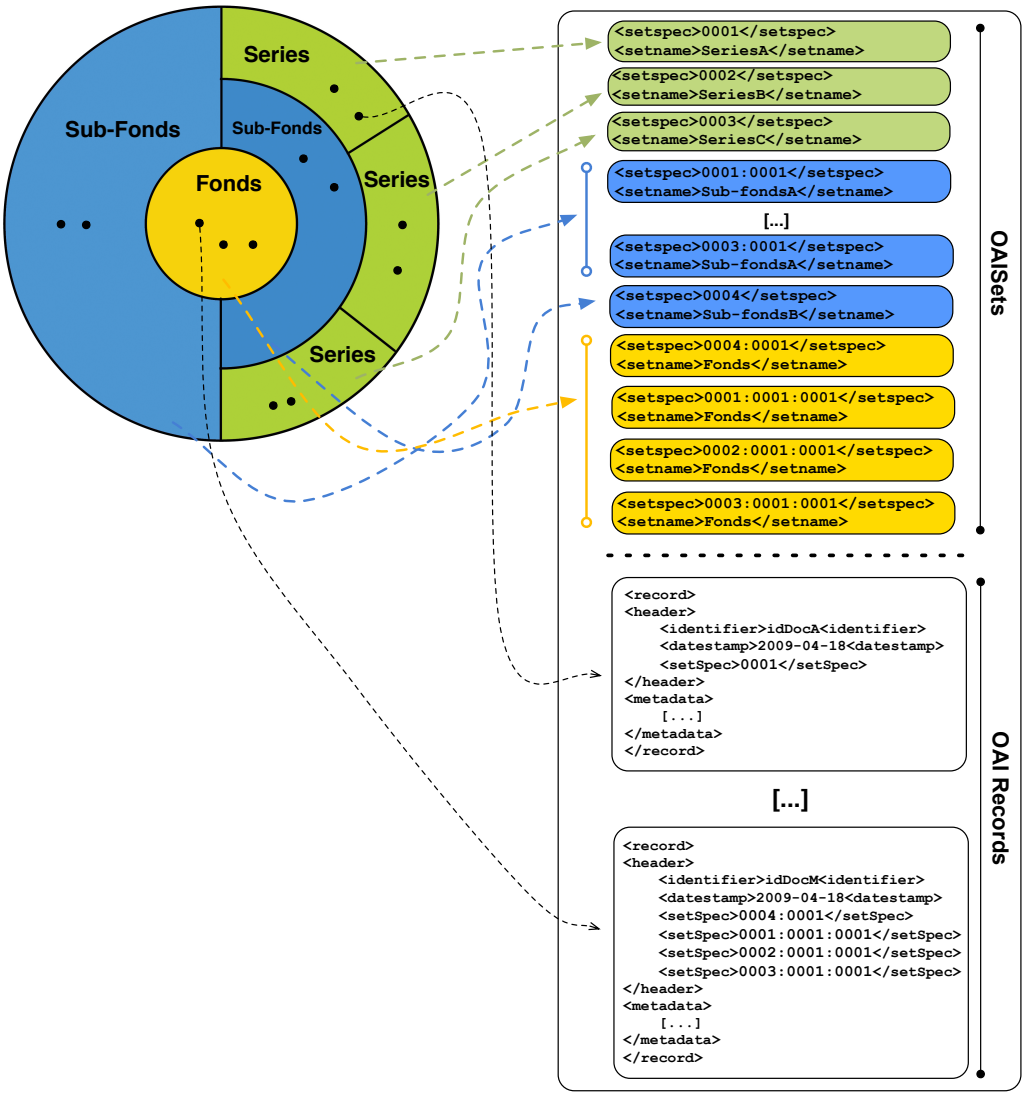


Figure 7.9: A sample archive represented throughout the INS-M and mapped into OAI-Sets and OAI Records.

7.4.2 Access, Exchange and Manipulate an Archive Through Sets

By means of the set-based application we have just described we dispose of a variable granularity access to the structure and to the content of an archive. Indeed, each OAI-Set is individually accessible as well as each single metadata. From an OAI-Set we can easily reconstruct the relationships with the other OAI-Sets by exploiting the `setspec`; from a metadata we can reconstruct the relationships with the other metadata thanks to the membership information contained in their header.

Throughout the OAI-Sets and Dublin Core metadata – that we know can be replaced by other metadata formats or by an extension provided by means of a Dublin Core Application Profile – approach we can easily use OAI-PMH to exchange a single set or a single metadata, thus allowing a *variable granularity exchange*. Furthermore, from the identifier of an OAI-Set

we can reconstruct the hierarchy through the ancestors to the root. By means of OAI-PMH it is possible to exchange a specific part of the archive while at the same time maintaining the relationships with the other parts of it. The *NS-M fosters the reconstruction of the lower levels of a hierarchy*; thus, with the couple NS-M and OAI-PMH applied to the archive, if a harvester asks for an OAI-Set representing for instance a sub-fonds it recursively obtains all the OAI-subsets and items in the subtree rooted in the selected sub-fonds.

When we consider the INS-M the point-of-view and the functionalities are somehow reversed; indeed, if a harvester asks for an OAI-Set representing for instance an archival series, it recursively obtains all the OAI subsets and records in the path from the archival series to the principal fonds that is the root of the archival hierarchy. The choice between a NS-M or INS-M should be made on the basis of the application context. For instance, often the information required by a user is stored in the external nodes of the archival tree [Shreeves et al., 2003]. If we model the archival tree by means of the INS-M, when a harvester requires an external node of the tree it will receive all the archival information contained in the nodes up to the root of the archive. This means that a Service Provider can offer a potential user the required information stored in the external node and also all the information stored in its ancestor nodes. This information is very useful for inferring the context of an archival metadata which is contained in the required external node; indeed, the ancestor nodes represent and contain the information related to the series, sub-fonds and fonds in which the archival metadata are classified.

The INS-M *fosters the reconstruction of the upper levels of a hierarchy* which in the archival case often contain contextual information which permit the relationships of the archival documents to be inferred with the other documents in the archive and with the production and preservation environment. We can see how the possibility of going from one set data model to the other by means of the defined mapping functions is very useful in the archival context; we can address the user requirements in the most effective way without being bound to the properties of a single model of choice. Furthermore, this application of the NESTOR Model allows us to adopt standard Digital Library technologies exploiting their native functionalities for innovative purposes without any changes in their basic functioning.

Lastly, we can see that this application is particularly well-suited for use in conjunction with the presented language techniques [Agosti et al., 2010a]. Indeed, the representation of an archive as an organization of sets and Dublin Core metadata makes it easier to determine the subject of each single metadata and thus to apply the "association to a class" solution; in the same way the metadata enrichment can be adopted because the Dublin Core metadata are well-suited to automatic processing. In this way the solutions proposed to enable cross-language access to digital contents can be applied also with the archival metadata, thus opening up these valuable resources to a significant service offered by the Digital Library technology.

7.4.3 Hierarchical Aggregations of Digital Resources

The NESTOR Model allows us to represent hierarchies of resources exploiting the flexibility and the adaptability of the set data models. We have seen that these models have been

proved to enable a variable granularity access and exchange of hierarchically structured metadata retaining their context and relationships. They can be straightforwardly applied to OAI-PMH extending the features of this widely-adopted protocol without any changes in its internal functioning [Ferro and Silvello, 2008b]. Now, we show how we can deal with hierarchical structured resources considering both metadata and digital objects. OAI-ORE has been proven to be a well-suited solution for identifying, accessing and exchanging aggregations of digital resources and it is a standard Digital Library technology [Doerr et al., 2010; VV. AA., 2010].

We have described the OAI-ORE Data Model in Section 5.3 but in order to show how the NESTOR Prototype makes use of OAI-ORE, we introduce a formal and more compact representation of its entities.

We indicate with $UA \subseteq H = \{ua_1, \ldots, ua_k, \ldots, ua_w\}$ the set of URI-A identifying the Aggregations and with $\eta_A : UA \rightarrow R$ the restriction of η[8] $(\eta|_A)$ to UA; the image of η_A is the set of Aggregations $A \subset R = \{a_1, \ldots, a_k, \ldots, a_w\}$. In the same way, we indicate with $URM \subseteq H$ the set of URI-RM identifying the Resource Maps and we define $\eta_{RM} : URM \rightarrow R$ to be the restriction $\eta|_{RM}$ where $RM \subset R$ is the set of Resource Maps. Finally, we indicate with $UAR \subseteq H$ the set of URI-AR identifying the Aggregated Resources. We define $\eta_{AR} : UAR \rightarrow R$ to be the restriction $\eta|_{AR}$ where $AR \subset R$ is the set of Aggregated Resources. Every $rm_i \in RM$ must describe one and only one $a_j \in A$, but a_j may be described by more than one Resource Map; thus, we indicate with $\varphi_{RMA} : RM \rightarrow A$ a surjective function which maps every Resource Map to the Aggregation it materializes. Every $ar_i \in AR$ may be aggregated by more than one $a_j \in A$.

We indicate with $UP \subseteq H = \{up_1, \ldots, up_k, \ldots, up_z\}$ the set of URI-P identifying the Proxies. We define $\eta_P : UP \rightarrow R$ to be the restriction $\eta|_P$ where $P \subset R$ is the set of Proxies. We indicate with $\varphi_{PAR} : P \rightarrow AR$ a function which maps a Proxy to the Aggregated Resource *for which* it is a proxy and with $\varphi_{PA} : P \rightarrow A$ a function which maps a Proxy to the Aggregation *in which* it is a proxy.

The *Nested Aggregations* feature enables the definion of Aggregations of Aggregations; this is consistent in the OAI-ORE data model because an Aggregation is a Resource which can also be seen as an Aggregated Resource of another Aggregation. Thanks to this feature, a partial order exists between Aggregations, call it $\prec_a$; more formally: for all $\{a_i, a_j\} \in A$ we say that $a_i \prec_a a_j$ if and only if the Aggregation a_i is aggregated by a_j. Now we can formally define the concept of *OAI-ORE Model*.

Definition 7.2.
Let $\mathcal{E} = \{A, R, AR, P, UA, UR, UAR, UP\}$ be the collection of OAI-ORE entity sets and $\Phi = \{\eta_A, \eta_{RM}, \eta_{AR}, \eta_P, \varphi_{RMA}, \varphi_{PAR}, \varphi_{PA}\}$ be the set of OAI-ORE functions. We define $ORE = \langle \mathcal{E}, \Phi \rangle$ to be an OAI-ORE Model.

From the definition of the OAI-ORE Model we can define a formal relation between it and the NESTOR Model; in order to define a mapping between a family of sets $\mathcal{F}_I$ defined in the

[8] The function η is defined in Definition 7.1.

NESTOR Model and an OAI-ORE Model $ORE = \langle \mathcal{E}, \Phi \rangle$ we have to take into account the two main entities in the NESTOR MODEL which are: sets and the resources belonging to them.

The basic idea is that every set $F_j \in \mathcal{F}_I$ becomes an aggregation $a_j \in A$ and consequently, every resource $r_t \in R$ belonging F_j becomes an aggregated resource $ar_t \in AR$ aggregated by a_j.

Theorem 7.1. *Let $\mathcal{F}_I$ be a family of sets and $ORE = \langle \mathcal{E}, \Phi \rangle$ an OAI-ORE model. Let $\gamma : \mathcal{F}_I \rightarrow A$ be a bijective function such that $\forall F_j \in \mathcal{F}_I, \exists! a_j \in A \mid \gamma(F_j) = a_j \Rightarrow \gamma^{-1}(a_j) = F_j$.*

Then $\forall F_j, F_k \in \mathcal{F}_I \mid F_j \subseteq F_k \Rightarrow \exists! a_j, a_k \in A \mid a_j \prec_a a_k$.

Proof. We know that every element belonging to $F_j \in \mathcal{F}_I$ is a resource $r_t \in R$, thus if $\gamma(F_j) = a_j \in A$, r_t is represented as ar_t aggregated by a_j. Ab absurdo suppose that $\exists F_j, F_k \in \{F_i\}_{i \in I} \mid F_j \subseteq F_k \Rightarrow \gamma(F_j) = a_j \not\prec_a \gamma(F_k) = a_k$. This means that $ar_t \in AR$ exists such that ar_t is aggregated by a_j but not by a_k and this implies that $\exists r_t \in R \mid r_t \in F_j \wedge r_t \notin F_k \Rightarrow F_j \not\subseteq F_k$. □

From Theorem 7.1 - valid for both a NS-F and a INS-F - we can see that every set in a family $\mathcal{F}_I$ is mapped into an Aggregation in the OAI-ORE model; the inclusion order between the sets is maintained by the partial order defined between the nested Aggregations of OAI-ORE. Then, by the means of the function φ_{RMA} a Resource Map is associated with each Aggregation. Every resource belonging to a set $F_j \in \mathcal{F}_I$ is mapped into an Aggregated Resources belonging to the Aggregation mapped from F_j. Thus, we can map a NS-F or a INS-F into a correspondent OAI-ORE model making it possible to identify, access and exchange a hierarchy of resources as an aggregation on the Web.

From Figure 7.10 we can see how the NS-F representing the archive described in Example 7.2 is straightforwardly mapped into an OAI-ORE model exploiting the defined functions; in order to enhance the differences between metadata and digital objects we represent these elements with different shapes in the NS-F – i.e. a square for the metadata and a circle for the digital objects. We can see how every set is mapped into an Aggregation and every digital resource, represented as an item belonging to the sets, is mapped into an Aggregated Resource. For space reasons in this figure we do not report all the Aggregated Resources belonging to the Aggregations. In this figure we call "$A_1, \ldots A_5$" the sets of the NS-F representing the archive, in order to create a direct association between them and the aggregation in OAI-ORE.

The OAI-ORE model permits us to define a *further semantic layer over the Aggregated Resources*; indeed, in the NS-F both digital objects and metadata are items belonging to the sets but in the OAI-ORE model we can define different kinds of relationships between the Aggregated Resources using the Proxies. For instance, in Figure 7.10 we represented two Proxies p_1 and p_2 related by the relationships "`xyz:isMetadataOf`"; thus, throughout p_1 and p_2 we can say that the Aggregated Resource ar_1 is the metadata describing the digital object ar_2. In Figure 7.10 ar_1 is metadata m_1 in the NS-F and ar_2 is the digital object do_1.

By means of the set data models defined by the NESTOR Model we can associate a *handle* to each resource in the archival hierarchy and thus access and exchange every metadata or

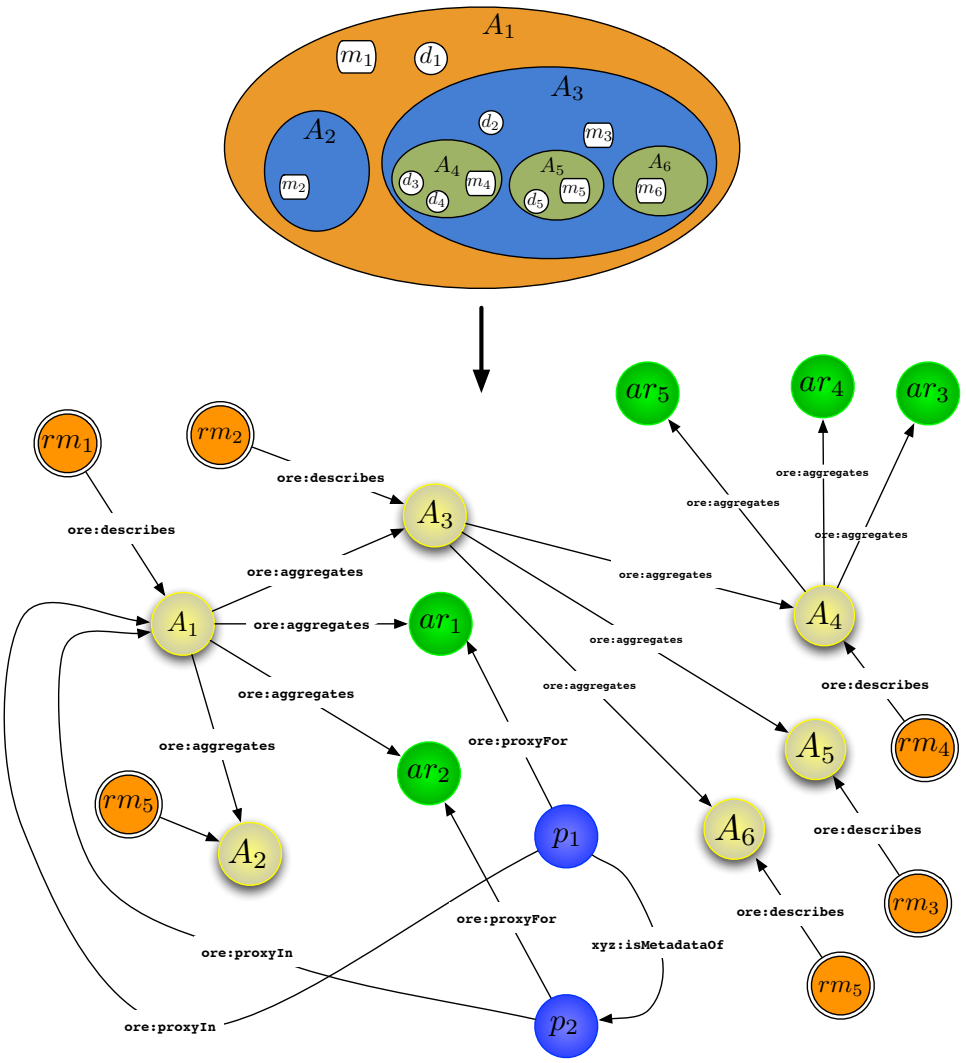


Figure 7.10: How it is possible to map archival resources into an OAI-ORE Model.

digital object with a variable granularity. At the same time, the archive is represented as an aggregation and its structure is preserved by the inclusion dependencies in the NESTOR Model and by a partial order between aggregations in the OAI-ORE Model. In order to relate metadata with the digital objects they describe, we do not have to use different metadata (e.g. METS) or to modify the metadata embedding a hard-coded URL in the digital objects; all the relationships are expressed throughout RDF triples defined in the Resource Maps without implying any modification of the digital object content. Furthermore, the NESTOR Model and OAI-ORE allow us *to define new logical organizations without any duplication of the digital objects and to manage them.*

The set-based application we presented provides a general solution for managing, accessing and exchanging hierarchies of digital resources with a variable granularity while retaining their context and relationships.

7.5 SIAR: A User-Centric Digital Archive System

The SIAR (*Sistema Informativo Archivistico Regionale*) is a project supported by the Italian Veneto Region which aim is to design and develop a Digital Archive System. The main goal of the SIAR is to develop a system for managing and sharing archive metadata in a distributed environment. Archives are geographically distributed across the Veneto Region and thus the archival resources they preserves are preserved in several local archives; the SIAR objective is to develop an information system able to create, manage, access, share and provide advanced services on archival metadata [Agosti et al., 2007c].

The design and development of the SIAR system rely on the NESTOR Framework; indeed, the NESTOR Model is adopted to model and represent the archives and the archival resources, and the set-based application envisioned in the NESTOR Prototype which provides a concrete instantiation for a Digital Archive System. We can say that the SIAR system is the last building block of the NESTOR Framework representing a kind of "*NESTOR Prototype in action*".

The description of the SIAR system allows us to give a deep insight of a possible concrete implementation of the NESTOR Prototype. Furthermore, it permits us to take users into account; indeed, all the design and development phases of the SIAR system have been characterized by a continuous feedback between computer scientists developing the system and the archivists that are interested to use it. The continuous feedback with the users allows us to add a further level of analysis of the NESTOR Framework; in fact, the users have been involved in both components of the framework: the Model and the Prototype.

We have considered the software engineering practice in order to point out six main phases that characterize the development of the SIAR system. In Figure 7.11 we can see the collaboration between computer scientists and archivists in the center of the six main phases that brought to the realization of the SIAR system. At the same time, it represents also the continuous feedback with the users in each one of the six phases:

Ideation: we have defined the goals of the project and the direction that it has to follow. This phase defines also the way in which the project is carried on – e.g. how archivists and computer scientists have to work together or who are the users of the system. In this phase the very nature of the archives and archival descriptions has been taken into account and we have analyzed the state of the art of digital archives.

Analysis of Requirements: in this phase we have defined the minimum set of requirements that the SIAR system has to fulfill in order to meet the archivists and general user needs.

Design: in this phase we have set the content and functional configuration parameters of the SIAR system defining the resources that are exploited by the system and specifying aspects of the system functionality perceived by the end-users. Together with the users we have designed the metadata schemas – i.e. Dublin Core Application Profiles – to describe the archival resources and that have to be managed by the SIAR system.

Data Model: in this phase we have defined the NESTOR Model. We have discussed with the

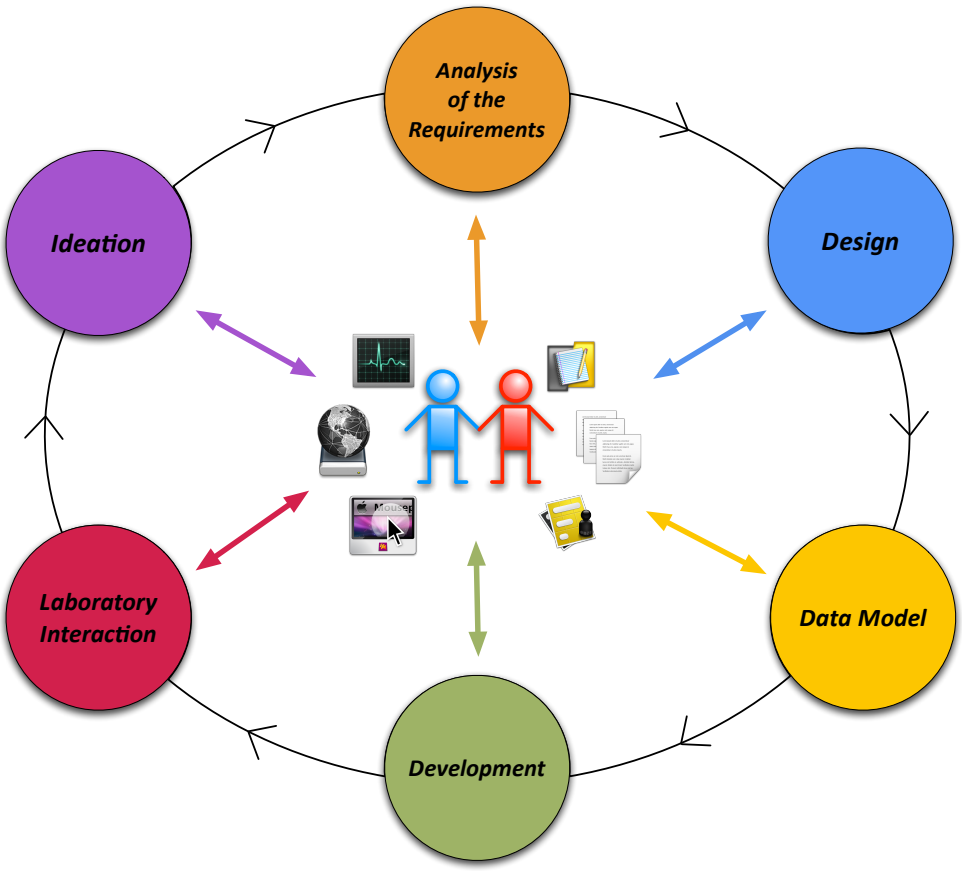


Figure 7.11: The six main phases carried out within the SIAR system.

archivists the functionalities and the possibilities of these set data models defining an innovative methodology to model the archives and the archival descriptions.

Development: in this phase we have instantiated the NESTOR Model adopting standard Digital Libraries technologies well-suited to meet the archival requirements.

Laboratory interaction: in this phase the SIAR system has been tested and its functionalities are tried by the archival users. Their suggestions and critiques are taken into account to understand which requirements are satisfied by the SIAR system and where it needs to be revised.

The ideation and the analysis of the requirements have been deeply analyzed in this chapter where we pointed out how the set-based instantiation of the NESTOR Model meets the requirements that also the SIAR system has to accomplish.

The design phase has been crucial to maintain the system aligned with respect to the information and functional needs of its end-users. The main purpose of the design phase is to set the content and functional configuration parameters of the SIAR system. The former parameters define the resources that are exploited by the system, like repositories of content, ontologies,

classification schemas and authority files. The latter parameters specify aspects of the system functionality perceived by the end-users like, for example, the result set format, the query language, the user profile formats, the document model [Candela et al., 2007a].

The work between computer scientists and archivists has been fundamental to define a trade-off between the technological possibilities and constraints, and the archival necessities. A consistent part of the work focused on the definition of the metadata formats for archival descriptions and for production and preservation subjects – i.e. authority files. So, in the SIAR system together with the archivists we designed an extensible metadata format for the archival description which relies on the Italian catalog of archival resources [Vitali, 2010] developed in the context of the *Catalog of Archival Resources* within the National Archival Portal[9]. This choice allows us to use different kinds of metadata formats and at the same time to export the SIAR metadata towards the National Archival Portal. In this way we set the ground for the use of a well-defined and wide adopted metadata format that at the same time can encompass most of all the necessities of the archivists.

In the rest of the work we dedicate further attention to the last three phases; indeed, we analyze the architecture of the SIAR system and then, we present the details of the implementation of the data model based on the set-based instantiation of the NESTOR Model. Lastly, we report the results of the user study we conducted on the SIAR system that points out the advantages of the NESTOR Framework as well as some issues that we have to address for what is concerned with the user functionalities and graphical interfaces.

7.5.1 The Architecture of the SIAR

The architecture designed for the SIAR system is divided into three basic layers: the data exchange infrastructure [Agosti et al., 2007b; Ferro and Silvello, 2008a], the metadata management layer [Ferro and Silvello, 2009a; Silvello, 2008] and the user interfaces layer; we can see a graphical representation of the SIAR architecture in Figure 7.12.

The transportation layer is based on the OAI-PMH protocol permitting the metadata exchange between the local archives spread across the territory. In the SIAR system, the Veneto Region is the Service Provider which provides advanced services such as data and public access to the harvested descriptive metadata and the archive keepers act as Data Providers because they supply descriptive metadata. Local archives maintain the control over their archives and maintain the property of their documents; on the other hand the Veneto Region constitutes a central repository where the user community can consult all the archives of the territory. When we consider the authority files the transport layer acts in the opposite way. Indeed, the Veneto Region assumes the role of the authority control composing the authority archive and sharing the authority files with the local archives. The authority files are access points either for the descriptive metadata harvested by the Service Provider of the Veneto Region and for the descriptive metadata retained by the local archives. Local archives act as Service Providers when

[9]The National Archival Portal is a project developed by the Directorate General of the Ministry of Cultural Assets and Activities. `http://www.archivi.beniculturali.it/`

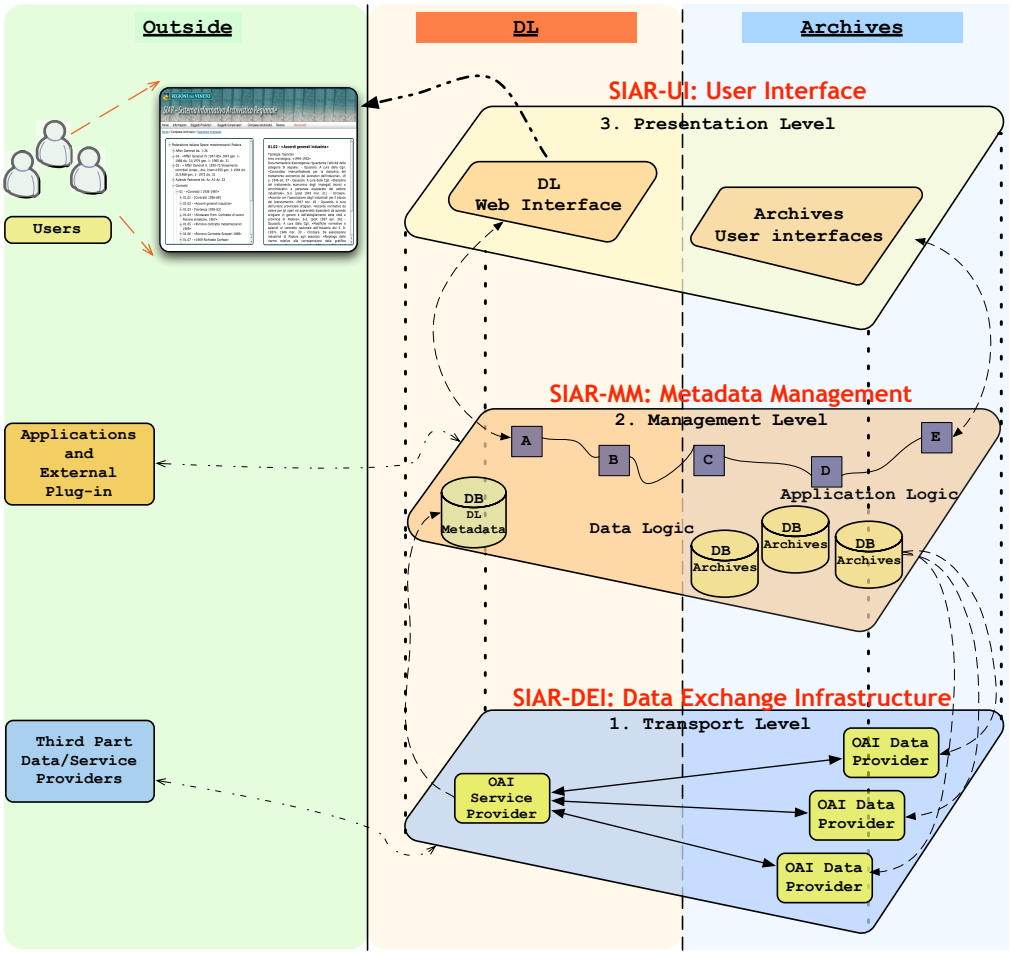


Figure 7.12: The Architecture of the SIAR System.

they harvests the authority files from the Veneto Region that in this case acts as a Data Provider. This approach permits us to concentrate the resources required to produce the authority files in the Veneto Region without replicating the effort in the local archives.

The metadata management layer is the central component of the system architecture. By means of this level it is possible to manage, preserve, retrieve and share full expressive archival metadata. It is composed by the database, the data logic and the application logic. The data logic is realized by a component called *datastore*, whereas the application logic is composed by the *service manager* and the *Web component manager*. We defined four main entities in the SIAR system: metadata, set, user and group; the main function of the database and the datastore is to create, read and update the various instances of these entities. Furthermore, they supply the data to the application logic. The service manager realizes the various services of the SIAR system such as the representation of metadata, the reconstruction of the set organizations and the OAI-PMH data and service providers. The Web component manager implements those methods that enable the interaction of the services with the Web services by exploiting the SIAR. Currently we have designed a Web service that realizes a user interface interacting with

the SIAR. The main role of this layer is to define the set organizations maintaining the archival hierarchies by following the rules defined in the NESTOR Model. Another function of the data management layer is to provide a mechanism that enables the local organizations to describe their archives natively as an organization of nested sets and Dublin Core metadata.

The third level of the SIAR architecture is the presentation layer constituted by the user interfaces. The system presents two main interfaces: the first is a general-purpose interface dedicated to a generic user-type such as archivists, historical researchers, public administrations or private organizations that will use the advanced services available in the SIAR; the second is dedicated to specialized users who can use this interface to describe an archive as well as to add, remove or update archival metadata.

To this purpose, the SIAR system defines two user roles: archival users and general users. The former one can create, modify and delete the metadata, instead the second one can only consult the metadata in the system. The archival part of the user interface provides the users with several forms where it is possible to insert and modify the archival metadata. These forms are shaped on the basis of the metadata schemas that have been designed. Together with the archivists we defined some visual aids to help the user in the insertion of the archival descriptions – e.g. instructions about how to compile the fields of the forms, a graphical representation of the inserted archives where it is possible to add archival divisions to the archival hierarchy or to add descriptions to a specific division. The insertion of new archival descriptions is guided by the system; for instance, if the root of the archive is a “fonds” the children of this node must be “sub-fonds” or a “series” but it cannot be another “fonds”. We developed several controlled vocabularies to guide the users in the description process.

7.5.2 The Metadata Management Layer

The SIAR metadata management layer is the central component of the SIAR System. Throughout this level it is possible to manage, preserve, retrieve and share full expressive archival metadata.

In Fig. 7.13 we can see a sketch of the 3-layers architecture with a zoom on the metadata management level composed by: the database, the data logic and the application logic. The data logic is realized by a component called *datastore*, instead the application logic is composed by the *service manager* and the *Web component manager*.

We can clearly see the four main entities of the SIAR: metadata, set, user and group; the main function of the database and the datastore is to create, read and update the various instances of these entities. Furthermore, they supply the data to the application logic. The service manager realizes the various services of the SIAR such as the representation of metadata, the reconstruction of the set organization and the OAI-PMH data and service providers. The Web component manager implements those methods that permit the interaction of the services with the Web services exploiting the SIAR. Currently we have designed a Web service that realizes a user interface interacting with the SIAR; in Figure 7.13 it is represented as the presentation logic. In Figure 7.14 we can see a screenshot of the user interface of the SIAR system; in

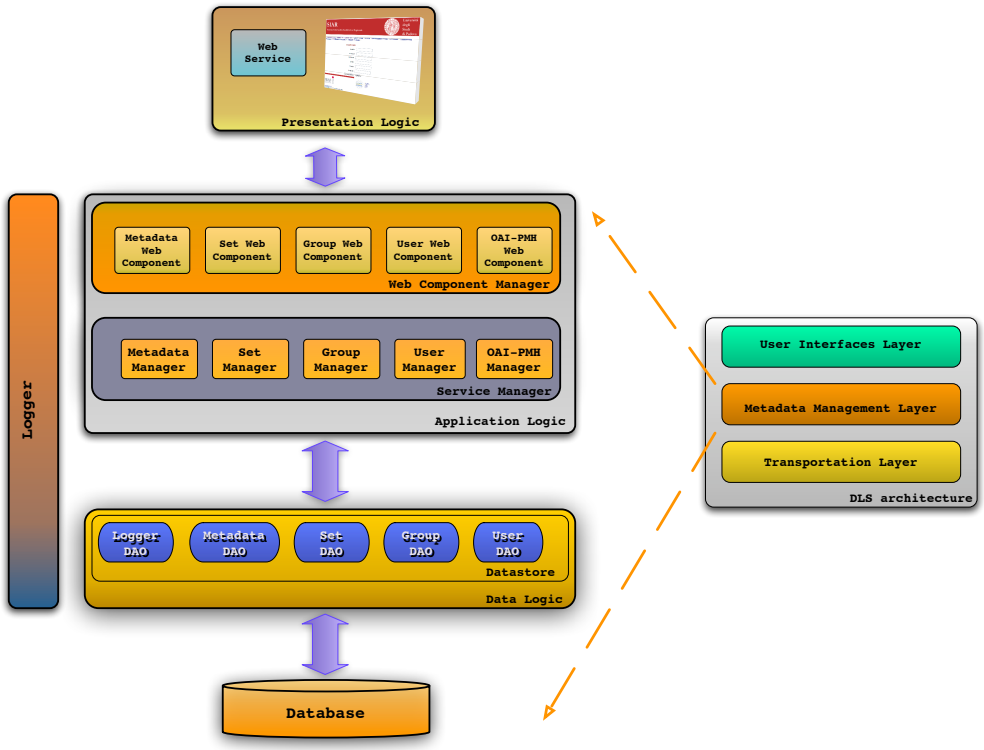


Figure 7.13: Composition of the SIAR metadata management layer

this screenshot we show the graphical interface through which a user can consult the archival descriptions in the SIAR system.

The second component presented in the Fig. 7.13 is the logger. The logger keeps track of all the activities of the system at all the levels; indeed, it is the only component transversal to the whole metadata management layer. The logger registers all the accesses of the users and all the operations done by the system such as the creation of metadata, sets, users or groups, the access and any update of the entities or the exceptions risen and handled by the system.

The Database Conceptual Schema

The conceptual design of the database reflects the world that the SIAR represents. In Fig. 7.15 we can see the conceptual schema of the database.

The conceptual schema is composed of four main entities: metadata, set, user and group. The metadata entity is defined by five attributes: *id*, this is the unique identifier of a metadata; the identifier is assigned automatically by the system, calculating an hash function of the metadata content. *body* containing the metadata content encoded in XML; the *body* is defined in the database as a native XML data type. Every metadata format encoded in XML can be stored in the database and the *schema* attribute reports the metadata format; throughout this attribute we are able to individuate the metadata format and to parse the XML file in a correct way. *Created* which is the attribute storing the time stamp in which the metadata was created in the database

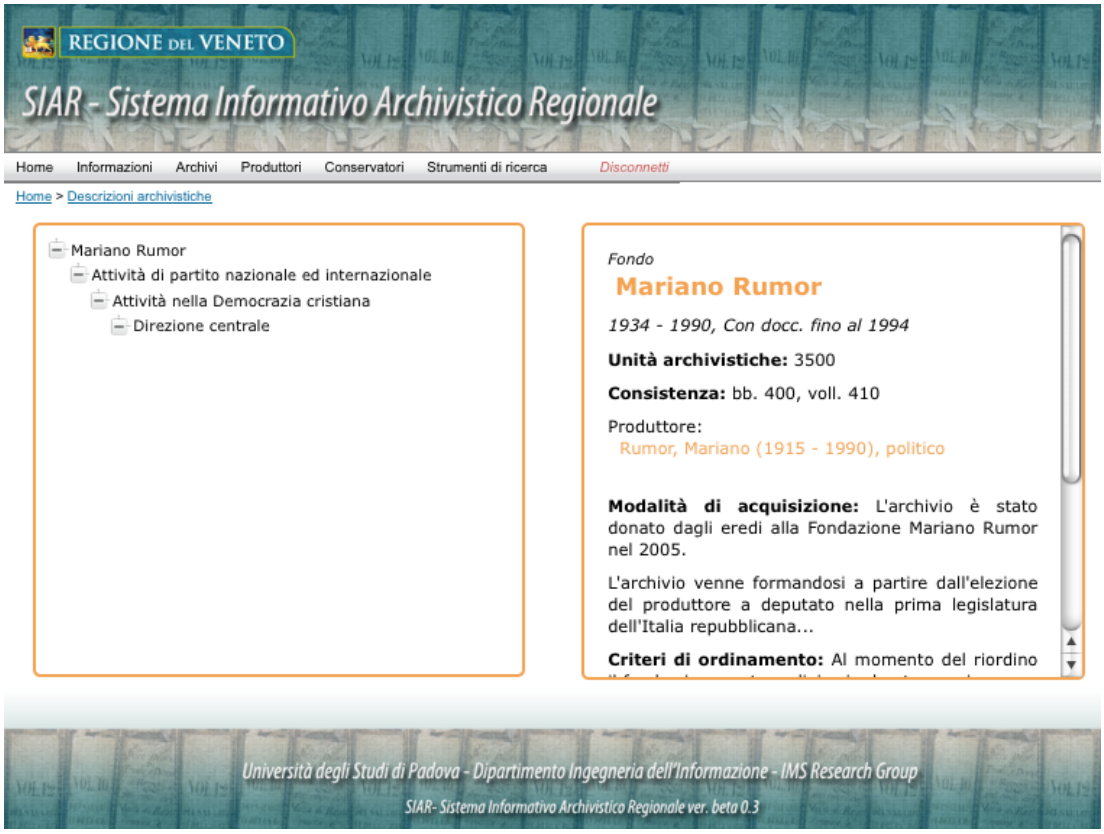


Figure 7.14: The SIAR user interface: a screenshot of the interface for the consultation of the archival descriptions.

and *updated* which stores the time stamp in which a metadata could have been updated.

The recursive relationship called *annotate* indicates if a metadata annotates (*annotator*, the metadata is the annotator of other metadata) some other metadata or if it is annotated (*annotation*) by other metadata. Thanks to this relationship we can have notes on metadata expressed as metadata themselves, we can retain the metadata history preserving all the versions of metadata that have been modified and we can retain information about the original repository of the metadata if they have been harvested by means of the OAI-PMH protocol. A metadata can annotate many other metadata, for instance in the *history case* an old metadata is the annotator of its updated version and the *repository case* where the metadata describing a repository is the annotator of all the metadata coming from that repository. Furthermore, annotate recursive relationship is useful to represents nested metadata. Metadata are the most important entities in the SIAR system and they establish relationships with all the other entities. The *author* relationship specifies that a metadata must be created by a user also if it has been harvested via OAI-PMH protocol. The *belong* relationship indicates that a metadata can belong to a specific set or to many sets defined in the system; a metadata can also belong to no set at all. The *access* relationship indicates that each metadata must have at least one group with read or write permissions on it (e.g. a metadata has to be read at least by the group to which the creator user belongs) and can have many groups with permissions on it.

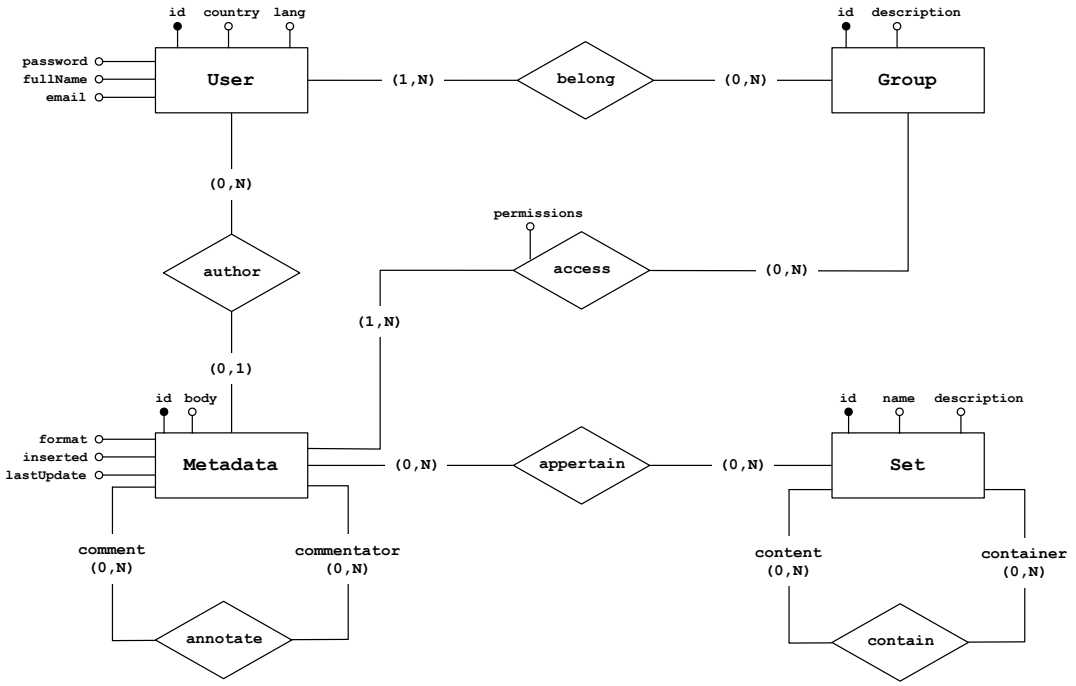


Figure 7.15: SIAR Conceptual Database Schema

The *set* entity is defined by three attributes: *id* which is the unique identifier of a set, *name* which is a mandatory attribute indicating the name of the set and an optional attribute called *description* which can contain a free text description of the set. There is a recursive relationship called *contain* which indicates if a set is contained (it is a subset) by other sets or if it contains (it is a superset) other sets or both.

The *user* entity is defined by six attributes: *id* which is the unique identifier of a user that can be seen as the username to access the system, *password* the password chosen by the user to access the system, *fullname* contains the full name of the registered user, *email* is the e-mail address of the user, *country* and *lang* indicate the origin of the user and are useful for setting up multilingual services. The *group* entity is defined by an *id* and a *description* of the group. The group and the user entities are related by the *contain* relationships that establish to which groups a user belongs.

The Data Logic Component

The data logic component of the metadata management layer is constituted by the SIAR datastore that defines all the methods that the application logic may call on the data logic of the SIAR system. The SIAR datastore is independent from any particular DBMS and is composed of several components called *Data Access Objects* (DAOs). The DAOs are used to abstract and encapsulate all access to the database; the DAO manages the connection with the database to obtain and store data. Essentially, the DAOs act as adapters between the components and the database. In Fig. 7.13 we can see that the data logic is composed by five DAOs: the metadata

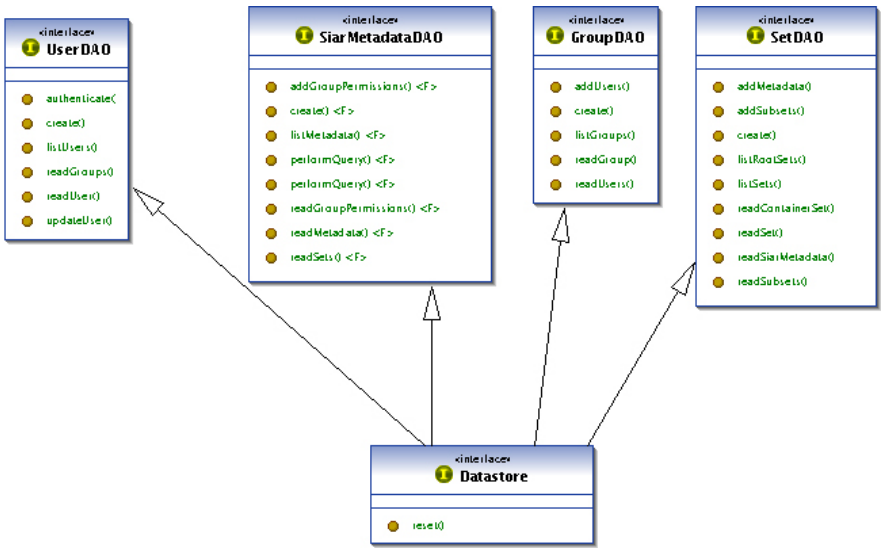


Figure 7.16: DLS Data Logic: The SIAR Datastore

DAO, the set DAO, the user DAO, the group DAO and the logger DAO. Every single DAO defines all of the methods that have to be provided for managing the corresponding entity in the database.

In Fig. 7.16 we can see the methods defined in the five DAOs of the SIAR datastore. For instance the Metadata DAO implements methods for creating or reading metadata, for adding a metadata to a set, for listing all the metadata belonging to a set, for adding read and write group permission or for reading all the set to which a specific metadata belongs. In the same way Set DAO implements methods for creating a set and adding and reading metadata from the sets. Furthermore, Set DAO defines several methods for obtaining the hierarchy of supersets of a set or the list of subsets.

The Application Logic

The application logic is constituted by two components: the service manager and the Web component manager. The service manager defines all the functionalities provided by the SIAR system; instead the Web components manager defines all the methods that provide support for the Web services implementation and elaborate external requests.

In Fig. 7.17 we can see how the service manager is composed: there is a service for each DAO in the datastore and an OAI-PMH service that implements the protocol functionalities. For instance the *user management service* provides the *authentication service* that permits the users authentication or the *reset password service* that allows a user to change his password.

The OAI-PMH service realizes the data and service provider features exploiting the other SIAR services. The data provider component of this service answers the requests of external service providers; for instance it creates lists of metadata encoded in XML files or returns the set organization of the system formatted as an OAI-PMH response. The service provider component enables the harvesting of metadata from other repositories and their storage in the

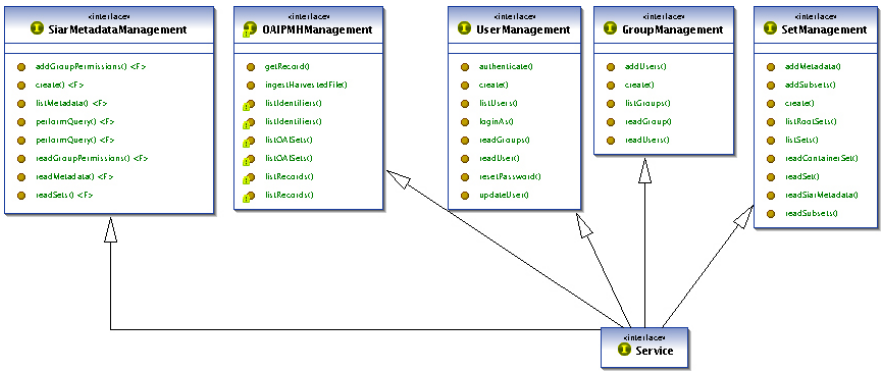


Figure 7.17: DLS Application Logic: The SIAR Service Manager

SIAR. In the same way external services can also be developed and added to the SIAR.

7.5.3 Laboratory Interaction

The laboratory interaction has been crucial to the development of the SIAR system because it has allowed us to verify if the archival requirements have been satisfied. The laboratory was conducted by a so-called "SIAR focus group" composed by:

- Archivists who work on proper metadata formats for archival descriptions.
- Computer scientists who work on the data model and system design and development.

Together archivists and computer scientists shaped the functionalities and the interactions with the SIAR system. In the user study conducted the users that used the SIAR system were asked to insert some archival descriptions about an archive in which they are working and also the metadata regarding the production and preservation subjects. Each step of the laboratory was characterized by a continuous feedback with the users.

The archival users were able to insert all their archival descriptions highlighting some relevant aspects related to the description policies that the SIAR project has to provide. The users have easily inserted several archival divisions exploiting the graphical aids provided by the user interface. They pointed out that the use of controlled vocabularies to help the insertion of the description is useful but at the same time it can be problematic. Sometimes archival descriptions have to go beyond the standard archival practice in order to describe some aspects of the archival reality that do not fit a standard model. For instance, there could be the necessity to create a sub-fonds as child of a series and the system should allow this possibility. Another important observation has regarded the definition of the authority files and their relationships with the descriptions; the users pointed out that the possibility to dispose of a manual mechanism to define the authority lists would be very useful to the archivists.

An important aspect highlighted by the users is the possibility to use different metadata formats to describe an archive and its related resources in the SIAR System. Indeed, in the

system we can find some metadata automatically imported from local archives and some metadata manually inserted by the archivists. These metadata are encoded by means of different metadata formats; indeed, the metadata which were imported automatically are encoded by simple Dublin Core and in some native formats adopted by the local systems, and the manually inserted metadata are encoded by means of the Dublin Core Application Profiles metadata defined in the context of the SIAR project. These aspects are transparent to the users; in fact, the user studies highlighted that the users did not have any problem in querying and consulting metadata encoded in different ways. The NESTOR Model has foremost impact in this aspect because it is not bound to specific technologies.

Currently, the SIAR system implements a user interface to describe an archive following a top-down descriptive procedure; this means that the description activity starts from the most general unit – e.g. a fonds – and proceeds all the way down through series and archival units. The users underlined the necessity to dispose of a bottom-up description mechanism allowing them to describe an archive starting from the lower units and then proceeding all the way up to the general fonds. Despite this important observation, the system currently offers several possibilities to manipulate the structure of the archive in a flexible way and without interfering with the archival metadata. Indeed, the users exploited the possibility to add newly defined archival divisions at any level of the archival hierarchy. For instance, it is possible to add a sub-fonds comprising several series even if this sub-fonds was not originally conceived in the first description of the archive.

The laboratory interaction with the users is a fundamental aspect that has to be taken into account in the context of Digital Libraries. The user studies we conducted in order to design, develop and test the SIAR system gave us useful insights not only for what is concerned with the user interfaces and the usability of the system, but also for what is concerned with the NESTOR Model. Furthermore, the laboratory interaction has been a fundamental step that allowed us to define the order of priorities for the future activities that have to be carried on in the research concerning the NESTOR Framework.

Chapter 8

Conclusions and Future Work

In this final chapter we summarize the main contributions of this work and discuss some future research directions.

8.1 Conclusions

In this work the *NEsted SeTs for Object hieRarchies (NESTOR) Framework* has been presented. This framework represents an effort lavished on two main complementary directions: the Model and the Prototype. The main aim was to define a new information and data model to deal with hierarchical organizations of resources; we levered on the wide field of Digital Libraries with a particular focus on the archives which retain meaningful resources modeled in a hierarchical way.

Particular attention was given to the tree model which is widely used in mathematics and in computer science and which represents the principal solution for modeling archival resources. We analyzed the use of the tree for this purpose and evaluated several technological solutions to represent, manage, access, exchange, manipulate and query hierarchically organized data and in particular archival resources, pointing out the advantages and issues of these approaches.

In the context of the NESTOR Framework we defined two set data models to represent hierarchically organized resources. These models are: the *Nested Sets Model* (NS-M) and the *Inverse Nested Sets Model* (INS-M). The foundational idea on which these models rely was given by the alternative graphical representations of the tree that in the literature are presented as aids to explain problems and algorithms related to the tree data structure. In particular, we focused on nested sets graphical representation of hierarchies, envisioning the possibility of defining set-based data models on its basis. To accomplish this goal, we formally defined the NS-M and the INS-M on a set-theoretical basis; this is the first difference with the tree which is instead defined on a graph-theoretical basis. Furthermore, we established the relationships between the NS-M and the INS-M, thus defining them as independent but complementary models.

The relationships between the set data models and the tree were mathematically defined by exploiting the bridges existing between set and graph theory. These formal definitions

allowed us to establish a common environment to deal with hierarchies where we can choose the most appropriate model between tree, NS-M and INS-M to represent the hierarchical reality of interest.

This formal basis allowed us also to define the relationships between the tree and the set data models properties and operations; in the NESTOR Model we can exploit the established formal relations to adopt the algorithms and the solutions defined for the trees and vice versa.

In this context the proved connections between the metric properties of the tree and of the set data models are meaningful results. Indeed, we formally defined and proved four proper distance measures for the NS-M and the INS-M. First-of-all, we defined the graphical distance and we proved the correspondence of the graphical distance between nodes in a tree and the graphical distance between sets in a family of sets defined within NS-M or INS-M.

The graphical distance is a measure that can be used within the same tree or family of sets but it does not give us a measure of similarity between two trees or between two families of sets. In order to accomplish this purpose we defined a content-based and a structure-based distance measure. The former is based on an extension to families of sets of Jaccard's coefficient. The latter is based on the joint use of the integer encodings defined to algebraically solve recursive queries in relational databases and the mathematical basic concepts exploited by bioinformatics to define algorithms to discover structures and patterns in biology. We proved these distances to be proper metrics for the NS-M and the INS-M and we provided a measure of similarity between two families of sets defined on a content-basis and on a structural-basis. Furthermore, we defined the NESTOR distance as a parametrized linear combination of the other two distances which allows us to weight the contribution of the content and the structural component.

These distance measures allowed us to point out a feature of the NESTOR Model that is the clear distinction between content and structural components. This feature gives us a further level of flexibility and expressiveness in dealing with hierarchies.

These results show us that the NESTOR Framework sets a common ground for dealing with hierarchies and that it is open to existing models, solutions and technologies; it exploits and enhances the state of the art in the fields of Digital Library and Database Systems, thus providing a further level of expressiveness and a theoretical common ground that can be exploited for the definition of innovative systems, functionalities and services. We exploited this formal basis to build a set-based application – i.e. the NESTOR Prototype – to deal with archival resources.

In the context of the NESTOR Prototype we showed how the NESTOR Model is an appropriate means for modeling an archive and its resources. We pointed out the expressive possibilities offered by the set data models that allow us to represent aspects of the reality which are not straightforwardly representable by means of the tree; for instance, the possibility for modeling both the archival structure by means of the nested sets and the archival resources by means of the elements belonging to them.

The NESTOR Prototype establishes a direct link between the NESTOR Model and standard Digital Library technologies such as: OAI-PMH, Dublin Core and OAI-ORE. We envisioned an application which meets the requirements defined by the dimensions of interoperability for

Digital Libraries which have to be respected also by a digital archive system; it is based on the joint use of the NESTOR Model and several Digital Library technologies. The advances of this set-based application are: the capacity of retaining the archival foundational characteristics, the ability of providing variable granularity access and exchange of archival resources – both metadata and full-content digital objects – and the possibility of manipulating and querying the archives and the archival resources.

This application found a concrete implementation in a Digital Archive System which is the *Sistema Informativo Archivistico Regionale* (SIAR). We presented this system, highlighting the peculiar aspects of the NESTOR Model and Prototype that have been exploited in the design and development phases. An outcome of the SIAR system is the possibility to exploit the advances of the NESTOR Framework together with the state-of-the-art models and technologies designed to deal with hierarchical structures and Digital Library resources. The flexibility of the NESTOR Framework is shown also by the possibility of adopting different metadata formats at the same time and within the same system, the capacity of manipulating the archival structure without interfering with the archival resources and vice versa, and the possibility of exploiting Digital Library technologies and services without any changes in their basic functioning.

The SIAR system and thus the NESTOR Framework represent a step forward with respect to the solutions proposed in the literature for dealing with archives. In particular, because they point out general solutions for dealing with archival resources where usually the solutions are provided on a case-by-case basis such as we showed happens when we consider an EAD-based application. Lastly, we showed how the SIAR system was designed and developed from a user-centric point-of-view, following the current trends envisioned by the Digital Library research community.

8.2 Directions for Future Work

The NESTOR Framework builds a solid base for future developments alongside its two main components: The Model and the Prototype.

We propose to enhance the *NESTOR Model* pursuing three new possible usages: its integration with other formal models defined in the context of Digital Libraries, its extension to envision new modeling possibilities, and its application in scientific fields other than the Digital Libraries. In particular, we envision the following paths for further research:

Modeling Annotations: In [Agosti and Ferro, 2007] a formal model for annotation was proposed; it defines digital annotations as independent information objects that can be organized in a hierarchical way. We envision the possibility of integrating this formal model with the NESTOR Model in order to provide a common formal basis for dealing with annotations in the context of Digital Libraries. This formal basis will then be exploited by the NESTOR Prototype to extend the proposed set-based application with the

possibility of accessing and exchanging digital annotations by means of OAI-PMH and implementing this extension in the SIAR system.

Extension of the Set Data Models: we propose to analyze further extensions of the NS-M and the INS-M in order to envision the possibility of modeling a wider spectrum of hierarchies. We want to analyze the possibility of relaxing some conditions in the definition of the NS-M and the INS-M. In a preliminary analysis we have seen that the INS-M can be extended to be an intersection structure [Davey and Priestley, 2002]. From this idea we can define extensions to model hierarchies that usually are represented by means of lattices.

XML Clustering: XML clustering widely exploits models and measures defined for the tree, such as tree edit distance and structural similarity measures. We propose to apply the NESTOR Model and the formally defined distance measures to this field in order to evaluate the impact of our proposed set-based approach.

We propose to further exploit the *NESTOR Prototype* envisioning the following further steps:

Digital Objects: As a consequence of the greater availability of digital objects in the archival domain we propose to extend the SIAR system with a concrete implementation of the aspects regarding the integration of the NESTOR Model within the OAI-ORE data model that we presented in Chapter 7.

Data Citation: We propose to define an automatic citation system for archival resources modeled by means of the NESTOR Model, evaluating the possibility to extend and improve the rule-based citation system [Buneman and Silvello, 2010] that we designed and developed for EAD-based applications.

A further research direction regards the possibility of defining a set of operations to manipulate and query the data represented by the NESTOR Model. At present, we envision an algebra which allows us to manipulate and query the structure and the content of the collection of sets defined within the NESTOR Model without flatting out the hierarchical relationships. The main directions we want to pursue are:

- To define an *extensible set of predicates* that allows us to impose constraints on the structure of the collections of sets and on the elements belonging to the sets.
- To define a *mechanism to identify the elements* on the basis of their values, the sets on the basis of the elements they contain and on the basis of the relationships they have with the other sets of the collection.
- To define *the operators to manipulate and query* the sets and their elements.

Bibliography

Abdi, H. (1990). Additive-Tree Representations. In *Lecture Notes in Biomathematics*, volume 84, pages 43–59. Springer, Heidelberg, Germany.

Abney, S., Flickenger, S., Gdaniec, C., Grishman, C., Harrison, P., Hindle, D., Ingria, R., Jelinek, F., Klavans, J., Liberman, M., Marcus, M., Roukos, S., Santorini, B., and Strzalkowski, T. (1991). Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In Black, E., editor, *HLT '91: Proceedings of the workshop on Speech and Natural Language*, pages 306–311, Morristown, NJ, USA. Association for Computational Linguistics.

Agosti, A., Ferro, N., and Silvello, G. (2010a). Enabling Cross-Language Access to Archival Metadata. In Cirinnà, C. and Lunghi, M., editors, *Cultural Heritage 2009: Empowering Users: An Active Role for User Communities*, pages 179–183.

Agosti, A., Ferro, N., and Silvello, G. (2010b). The NESTOR Framework: Manage, Access and Exchange Hierarchical Data Structures. In *Proceedings of the 18th Italian Symposium on Advanced Database Systems*, pages 242–253. Società Editrice Esculapio, Bologna, Italy.

Agosti, M., Albrechtsen, H., Ferro, N., Frommholz, I., Hansen, P., Orio, N., Panizzi, E., and Pejtersen, A. M.and Thiel, U. (2005). DiLAS: a Digital Library Annotation Service. In Boujut, J. F., editor, *Proceedings of the International Workshop on Annotation for Collaboration - Methods, Tools and Practices*, pages 91–101.

Agosti, M., Bonfiglio-Dosio, G., Ferro, N., and Silvello, G. (2008). Metodologie e percorsi interdisciplinari per la ideazione di un Sistema Informativo Archivistico. *Memoria dell'Accademia Galileiana in Scienze, Lettere ed Arti in Padova, Ente di Alta Cultura (D.P.R. 27/10/49 n.1005), Padova*.

Agosti, M., Brettlecker, G., Ferro, N., Ranaldi, P., and Schuldt, H. (2007a). Extending the DelosDLMS by the FAST Annotation Service. In Agosti, M., Esposito, F., and Thanos, C., editors, *Post-proceedings of the Third Italian Research Conference on Digital Library Systems (IRCDL 2007)*, pages 7–12. ISTI-CNR at Gruppo ALI, Pisa, Italy.

Agosti, M., Esposito, F., and Thanos, C., editors (2010c). *Digital Libraries - 6th Italian Research Conference, IRCDL 2010. Revised Selected Papers*, volume 91 of *Communications in Computer and Information Science*. Springer, Heidelberg, Germany.

Agosti, M. and Ferro, N. (2007). A Formal Model of Annotations of Digital Content. *ACM Trans. Inf. Syst.*, 26(1).

Agosti, M. and Ferro, N. (2010). Interoperabilità tra sistemi di biblioteche digitali. *DigItalia, Rivista del digitale nei beni culturali*, 5(1):95–112.

Agosti, M., Ferro, N., and Silvello, G. (2007b). An Architecture for Sharing Metadata among Geographically Distributed Archives. In Thanos, C., Borri, F., and Candela, L., editors, *Digital Libraries: Research and Development, First International DELOS Conference*, volume 4877 of *Lecture Notes in Computer Science*, pages 56–65. Springer, Heidelberg, Germany.

Agosti, M., Ferro, N., and Silvello, G. (2007c). Proposta metodologica e architetturale per la gestione distribuita e condivisa di collezioni di documenti digitali. *Archivi*, 2(2):49–73.

Agosti, M., Ferro, N., and Silvello, G. (2009a). Access and Exchange of Hierarchically Structured Resources on the Web with the NESTOR Framework. *Web Intelligence and Intelligent Agent Technology, IEEE/WIC/ACM International Conference*, pages 659–662.

Agosti, M., Jose Borbinha, J., Kapidakis, S., Papatheodorou, C., and Tsakonas, G., editors (2009b). *Proc. 13th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2009)*. Springer, Heidelberg, Germany.

Aho, A. and Ullman, J. D. (1992). *Foundations of Computer Science.* Computer Science Press, New York, New York, USA.

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1973). On Finding Lowest Common Ancestors in Trees. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, STOC '73, pages 253–265, New York, NY, USA. ACM.

Alexanderson, G. L. (October 2006). About the Cover: Euler and the Königsberg Bridges: A Historical View. *Bulletin (New Series) of the American Mathematical Society*, 43(4):567–573.

Alkhatib, R. and Scholl, M. H. (2009). Compacting XML Structures Using a Dynamic Labeling Scheme. In Sexton, A., editor, *Dataspace: The Final Frontier, 26th British National Conference on Databases, BNCOD 26, 2009*, volume 5588 of *Lecture Notes in Computer Science*, pages 158–170. Springer, Heidelberg, Germany.

Altman, M. and King, G. (March/April 2007). A Proposed Standard for the Scholarly Citation of Quantitative Data. *D-Lib Magazine*, 13(3/4).

Anderson, K. W. and Hall, D. W. (1963). *Sets, Sequences, and Mappings: The Basic Concepts of Analysis*. John Wiley & Sons, Inc., New York, NY, USA.

Apostolico, A. and Galil, Z. (1997). *Pattern Matching Algorithms*. Oxford University Press, Oxford, UK.

Bateson, P. P. G. and Hinde, R. A. (1976). *Growing Points in Ethology*. Cambridge University Press, Cambridge, UK.

Belkin, N. (1999). Understanding and Supporting Multiple Information Seeking Behaviors in a Single Interface Framework. In *Eighth DELOS Workshop: User Interfaces in Digital Libraries*, pages 11–18. European Research Consortium for Informatics and Mathematics.

Bender, M. A., Farach-colton, M., Pemmasani, G., Skiena, S., and Sumazin, P. (2005). Lowest Common Ancestors in Trees and Directed Acyclic Graphs. *J. Algorithms*, 57:75–94.

Biggs, N., Lloyd, E. K., and Wilson, R. J. (1986). *Graph Theory, 1736-1936*. Clarendon Press, Oxford, UK.

Bille, P. (2005). A Survey on Tree Edit Distance and Related Problems. *Theor. Comput. Sci.*, 337(1-3):217–239.

Borgman, C. L. (1999). What are Digital Libraries? Competing Visions. *Information Processind & Management*, 35(3):227–243.

Bountouri, L. and Manolis, G. (2009). Interoperability Between Archival and Bibliographic Metadata: An EAD to MODS Crosswalk. *Journal of Library Metadata*, 9(1/2):98–133.

Brooking, C., Shouldice, S., Robin, G., Kobe, B., Martin, J., and Hunter, J. (2009). Comparing METS and OAI-ORE for Encapsulating Scientific Data Products: A Protein Crystallography Case Study. In *Fifth IEEE International Conference on e-Science, 2009*, pages 148–155.

Buneman, P. (1971). The Recovery of Trees from Measures of Dissimilarity. In Kendall, D. and Tautu, P., editors, *Mathematics the the Archeological and Historical Sciences*, pages 387–395. Edinburgh University Press, Edinburgh, UK.

Buneman, P. (1974). A Note on the Metric Properties of Trees. *J. Combinatorial Theory (B)*, 17:48–50.

Buneman, P. (2006). How to Cite Curated Databases and How to Make Them Citable. In *Proc. of 18th Int. Conf: on Scientific and Statistical Database Management, SSDBM 2006*, pages 195–203. IEEE Computer Society.

Buneman, P. and Silvello, G. (September 2010). A Rule-Based Citation System for Structured and Evolving Datasets. *Bulletin of the Technical Committee on Data Engineering*, 3:33–41.

Buneman, P. and Tan, W. C. (2007). Provenance in Databases. In Chan, C. Y., Ooi, B. C., and Zhou, A., editors, *SIGMOD Conference*, pages 1171–1173. ACM Press, New York, USA.

Burkhardt, F. and Smith, S. (2001). *Correspondence of Charles Darwin, vol. VII (1858-59)*. Cambridge Press, Cambridge, UK.

Candela, L., Castelli, D., Ferro, N., Koutrika, G., Meghini, C., Pagano, P., Ross, S., Soergel, D., Agosti, M., Dobreva, M., Katifori, V., and Schuldt, H. (2007a). *The DELOS Digital Library Reference Model. Foundations for Digital Libraries (Version 0.96)*. ISTI-CNR at Gruppo ALI, Pisa, Italy.

Candela, L., Ioannidis, I., Koutrika, G., Ross, S., Schek, H. J., and Schuldt, H. (2007b). Setting the Foundations of Digital Libraries. *D-Lib Megazine*, 13(3/4).

Castelli, D., Candela, L., Manghi, P., Pagano, P., Tang, C., and Thanos, C. (2010). An Event-Centric Provenance Model for Digital Libraries. In [Agosti et al., 2010c], pages 79–88.

Cavalli-Sforza, L. L. (1997). Genes, Peoples, and Languages. *Proc. Natl. Acad. Sci. USA*, 94(15):7719–7724.

Cavalli-Sforza, L. L., Menozzi, P., and Piazza, A. (1994). *History and Geography of Human Genes*. Princeton University Press, Princeton, NJ, USA.

Cayley, A. (1891). On the Theory of Analytic Forms Called Trees. *Philos. Mag. 13, 19-30, 1857. Reprinted in Mathematical Papers*, 3:242–246.

Celko, J. (2000). *Joe Celko's SQL for Smarties: Advanced SQL Programming*. Morgan Kaufmann, San Francisco, California, USA.

Cheung, K., Hunter, J., Lashtabeg, A., and Drennan, J. (2008). SCOPE: A Scientific Compound Object Publishing and Editing System. *International Journal of Digital Curation*, 3(2).

Christophides, N. (1975). *Graph Theory - An Algorithmic Approach*. Academic Press, Inc., London, UK.

Christophides, V., Karvounarakis, G., Plexousakis, D., Scholl, M., and Tourtounis, S. (2004). Optimizing Taxonomic Semantic Web Queries Using Labeling Schemes. *Journal of Web Sematics 1, Issue*, pages 207–228.

Clarkson, K. L. (2006). Nearest-Neighbor Searching and Metric Space Dimensions. In Shakhnarovich, G., Darrell, T., and Indyk, P., editors, *Nearest-Neighbor Methods for Learning and Vision: Theory and Practice*, pages 15–59. MIT Press, Boston, MA, USA.

Cobena, G., Abiteboul, S., and Marian, A. (2002). Detecting Changes in XML Documents. In *Proceedings of the 18th International Conference on Data Engineering*, pages 41–52. IEEE Computer Society.

Cohen, P. J. (December 1963). The Independence of the Continuum Hypothesis. *Proc. Natl. Acad. Sci. USA*, 50(6):1143–1148.

Cook, T. (1993). The Concept of Archival Fonds and the Post-Custodial Era: Theory, Problems and Solutions. *Archiviaria*, 35:24–37.

Copson, E. T. (1968). *Metric Spaces*. Cambridge at the University Press, Cambridge, UK.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to Algorithms, second edition*. The MIT Press and McGraw-Hill, Boston, MA, USA.

Crestani, F., Vegas, J., and de la Fuente, P. (2004). A Graphical User Interface for the Retrieval of Hierarchically Structured Documents. *Inf. Process. Management*, 40(2):269–289.

Darwin, C. (1859). *The Origin of Species*. Barnes & Noble Classics, New York, New York, USA.

Davey, B. A. and Priestley, H. A. (2002). *Introduction to Lattices and Order - 2nd Ed.* Cambridge University Press, Cambridge, UK.

Diestel, R. (2006). *Graph Theory*. Springer, Berlin Heidelberg, Germany.

Doerr, M., Gradmann, S., Hennicke, S., Isaac, A., Meghini, C., and Van de Sompel, H. (2010). The Europeana Data Model (EDM). In *IFLA 2011: World Library and Information Congress: 76th IFLA General Conference and Assembly*, Gothenburg, Sweden.

Duranti, L. (1998). *Diplomatics: New Uses for an Old Science*. Society of American Archivists and Association of Canadian Archivists in association with Scarecrow Press, Lanham, Maryland, USA.

Durkheim, E. and Mass, M. (1902). De quelques formes de classification – Contribution á l'ètude des reprèsentations collectives. *Annèe sociologique*, VI:1–72.

Euler, L. (1736). Solutio Problematis ad geometriam situs pertinentis. *Commentarii Academiae Scientiarum Imperialis Petropolitanae*, 8:128–140.

Ferro, N. (2009). Annotation Search: The FAST Way. In [Agosti et al., 2009b], pages 15–26.

Ferro, N. and Silvello, G. (2008a). A Distributed Digital Library System Architecture for Archive Metadata. In Agosti, M., Esposito, F., and Thanos, C., editors, *Post-proceedings of the Forth Italian Research Conference on Digital Library Systems (IRCDL 2008)*, pages 99–104. ISTI-CNR at Gruppo ALI, Pisa, Italy.

Ferro, N. and Silvello, G. (2008b). A Methodology for Sharing Archival Descriptive Metadata in a Distributed Environment. In Christensen-Dalsgaard et al., B., editor, *Proc. 12th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2008)*, pages 268–279. Lecture Notes in Computer Science (LNCS) 5173, Springer, Heidelberg, Germany.

Ferro, N. and Silvello, G. (2009a). Design and Development of the Data Model of a Distributed DLS Architecture for Archive Metadata. In 5^{th} *IRCDL - Italian Research Conference on Digital Libraries*, pages 12–21. DELOS: an Association for Digital Libraries.

Ferro, N. and Silvello, G. (2009b). The NESTOR Framework: How to Handle Hierarchical Data Structures. In [Agosti et al., 2009b], pages 215–226.

Ferro, N. and Silvello, G. (2010). FAST and NESTOR: How to Exploit Annotation Hierarchies. In [Agosti et al., 2010c], pages 55–66.

Franklin, M. J., Moon, B., and Ailamaki, A., editors (2002). *Proceedings of the 2002 ACM SIGMOD International Conference on Management of Data*. ACM Press, New York, USA.

Fuhr, N., Tsakonas, G., Aalberg, T., Agosti, M., Hansen, P., Kapidakis, S., Klas, C. P., Kovács, L., Landoni, M., Micsik, A., Papatheodorou, C., Peters, C., and Sølvberg, I. (2007). Evaluation of Digital Libraries. *International Journal on Digital Libraries*, 8(1):21–38.

Furetière, A. (1690). *Dictionnaire universel contenant généralement tous les mots françois.* Arnout & Reinier Leers, Le Haye et Rotterdam.

Gallé, M. (2010). A New Tree Distance Metric for Structural Comparison of Sequences. In Apostolico, A., Dress, A., and Parida, L., editors, *Structure Discovery in Biology: Motifs, Networks & Phylogenies*, number 10231 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Germany.

Gilliland-Swetland, A. J. (2000). *Enduring Paradigm, New Opportunities: The Value of the Archival Perspective in the Digital Environment*. Council on Library and Information Resources, Washington, DC, USA.

Gödel, K. (1940). *The Consistency of the Continuum-Hypothesis*. Princeton University Press, Priceton, NJ, USA.

Goodrich, M. T. and Tamassia, R. (2001). *Data Structures and Algorithms in Java - 2nd Edition*. John Wiley and Sons, Inc., Hoboken, NJ, USA.

Goswami, A. (2008). *God is Not Dead: What Quantum Physics Tells Us About Our Origins and How We Should Live*. Hampton Roads Pub Co Inc, Charlottesville, Va, USA.

Gowers, T., Barrow-Green, J., and Leader, I., editors (2008). *The Princeton Companion to Mathematics*. Princeton University Press, Princeton, NJ, USA.

Gradmann, S. (2007). Interoperability of Digital Libraries: Report on the work of the EC working group on DL interoperability. In *Seminar on Disclosure and Preservation: Fostering European Culture in The Digital Landscape*. National Library of Portugal, Directorate-General of the Portuguese Archives, Lisbon, Portugal.

Guha, S., Jagadish, H. V., Koudas, N., Srivastava, D., and Yu, T. (2002). Approximate XML Joins. In [Franklin et al., 2002], pages 287–298.

Halmos, P. R. (1960). *Naive Set Theory*. D. Van Nostrand Company, Inc., New York, NY, USA.

Hanks, P., editor (1979). *Collins Dictionary of The English Language*. William Collins Sons & Co.Ltd., Glasgow, UK.

Harary, F. (1969). *Graph Theory*. Addison-Wesley Publishing Co., Boston, MA, USA.

Härder, T., Haustein, M., Mathis, C., and Wagner, M. (2007). Node Labeling Schemes for Dynamic XML Documents Reconsidered. *Data Knowl. Eng.*, 60(1):126–149.

Harel, D. and Tarjan, R. E. (1984). Fast Algorithms for Finding Nearest Common Ancestors. *SIAM Journal on Computing*, 13(2):338–355.

Haworth, K. M. (2001). Archival Description: Content and Context in Search of Structure. In Pitti, D. and Duff, W. M., editors, *Encoded Archival Description on the Internet*, pages 7–26. The Haworth Press, Inc.

Hayworth, K. M. (1993). The Voyage of RAD: From the Old World to the New. *Archiviaria*, 35:55–63.

Heil, G. (1991). *Corpus Dionysiacum: Pseudo-Dionysius Areopagita, De Coelesti Hierarchia, De Ecclesiastica Hierarchia, De Mystica Theologia, Epistulae*. Walter de Gruyter, Berlin.

Henze, H. R. and Blair, C. M. (1931). The Number of Structurally Isomeric Alcohols of the Methanol Series. *Journal of American Chem. Soc.*, 53(8):3077–3085.

Hrbacek, K. and Jech, T. (1999). *Introduction to Set Theory, 3rd Edition*. Marcel Dekker, Inc., New York. New York, USA.

International Council on Archives (1999a). ISAAR(CPF): International Standard Archival Authority Records (Corporate Bodies, Persons, Families), 2nd edition. Ottawa: International Council on Archives.

International Council on Archives (1999b). ISAD(G): General International Standard Archival Description, 2nd edition. Ottawa: International Council on Archives.

International Council on Archives – Committee of Best Practices and Standards (CBPS) (2008). ISDIAH: International Standard for Describing Institutions with Archival Holdings, 1st edition. Ottawa: International Council on Archives.

Ioannidis, Y. E. (2005). Digital Libraries at a Crossroads. *International Journal on Digital Libraries*, 5(4):255–265.

Ioannidis, Y. E., Maier, D., Abiteboul, S., Buneman, P., Davidson, S. B., Fox, E. A., Halevy, A. Y., Knoblock, C. A., Rabitti, F., Schek, H. J., and Weikum, G. (2005). Digital Library Information-Technology Infrastructures. *International Journal on Digital Libraries*, 5(4):266–274.

Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des alpes et des jura. *Bulletin del la Société Vaudoise des Sciences Naturelles*, 37:547–579.

Jech, T. (2003). *Set Theory*. Springer-Verlag, Berlin, Germany.

Kamfonas, M. (October/November 1992). Recursive Hierarchies: The Relational Taboo! *The Relational Journal*.

Kamke, E. (1950). *Theory of Sets*. Dover Publications, New York, NY, USA.

Kiesling, K. (2001). Metadata, Metadata, Everywhere - But Where Is the Hook? *OCLC Systems & Services*, 17(2):84–88.

Kim, J. (2004). EAD Encoding and Display: A Content Analysis. *Journal of Archival Organization*, 2(3):41–55.

Knuth, D. E. (1997). *The Art of Computer Programming, third edition*, volume 1. Addison Wesley, Reading, MA, USA.

Lagoze, C., Van de Sompel, H., Nelson, M. L., Warner, S., Sanderson, R., and Johnston, P. (2008a). Object Re-Use & Exchange: A Resource-Centric Approach. *CoRR*, abs/0804.2273.

Lagoze, C., Van de Sompel, H., P., J., Nelson, M., Sanderson, R., and Warner, S. (2008b). ORE Specification - Abstract Data Model. Technical report, OAI.

Lane, D. (2006). Hierarchy, Complexity, Society. In Pumain, D., editor, *Hierarchy in Natural and Social Sciences*, volume 3, pages 81–119. Springer, Berlin Heidelberg, Germany.

Lawrence, S., Lee Giles, C., and Bollacker, K. (1999). Digital Libraries and Autonomous Citation Indexing. *Computer*, 32(6):67–71.

Levergood, B., Farrenkopf, S., and Frasnelli, E. (2008). The Specification of the Language of the Field and Interoperability: Cross-Language Access to Catalogues and Online Libraries (CACAO). In Greenberg, J. and Klasore, W., editors, *DC-2008, Proc. of the Int'l Conf. on Dublin Core and Metadata Applications 2008*, pages 191–196. Universitätsverlag Göttingen, Germany.

Lipkus, A. (1999). A proof of the triangle inequality for the Tanimoto distance. *Journal of Mathematical Chemistry*, 26:263–265.

MacNeil, H., Wei, C., Duranti, L., Gilliland-Swetland, A., Guercio, M., Hackett, Y., Hamidzadeh, B., Iacovino, L., Lee, B., McKemmish, S., Roeder, J., Ross, S., Wan, W., and Zhon Xiu, Z. (2001). *Authenticity Task Force Report*. InterPARES Project: Vancouver, Canada.

Marx, K. and Engels, F. (1948). *Manifesto of the Communist Party*. Tribeca Books, New York, NY, USA.

Mikeal, A., Creel, J., Maslov, A., Phillips, S., Leggett, J., and McFarland, M. (2009). Large-Scale ETD Repositories: A Case Study of a Digital Library Application. In *Proceedings of the*

9th ACM/IEEE-CS joint conference on Digital libraries, Proceedings of the 9th ACM/IEEE-CS joint conference on Digital libraries, pages 135–144, New York, NY, USA. ACM Press.

Milton, J. (1674). *Paradise Lost, a Poem in Twelwe Books*. Random House Inc., New York, NY, USA.

Moore, G. (1982). *Zermelo's Axiom of Choice*. Springer-Verlag, Berlin, Germany.

Moreau, L., Groth, P., Miles, S., Vazquez-Salceda, J., Ibbotson, J., Jiang, S., Munroe, S., Rana, O., Schreiber, A., Tan, V., and Varga, L. (2008). The Provenance of Electronic Data. *Commununications of the ACM*, 51(4):52–58.

Moretti, F. (2005). *Graphs, Maps, Trees – Abstract Models for Literary History*. Verso, New York, NY, USA.

Na, G. and Lee, S. (2006). A Relational Nested Interval Encoding Scheme for XML Data. In Bressan, S., Kng, J., and Wagner, R., editors, *Database and Expert Systems Applications*, volume 4080 of *Lecture Notes in Computer Science*, pages 83–92. Springer Berlin Heidelberg, Germany.

O'Neil, P. E., O'Neil, E. J., Pal, S., Cseri, I., Schaller, G., and Westbury, N. (2004). ORDPATHs: Insert-Friendly XML Node Labels. In [Weikum et al., 2004], pages 903–908.

Otter, R. (1947). The Number of Trees. *Annals of Mathematics*, 49:583–599.

Pearce-Moses, R. (2005). *Glossary of Archival And Records Terminology.* Society of American Archivists.

Pepe, A., Mayernik, M., Borgman, C., and Van de Sompel, H. (2010). From Artifacts to Aggregations: Modeling Scientific Life Cycles on the Semantic Web. *JASIST (J. of the American Society for Inf. Science and Technology)*, 61:567–582.

Piazza, A. (2005). Evolution at a Close Range. In *Graphs, Maps, Trees*, pages 95–113, New York, NY, USA. Verso.

Pitti, D. and Duff, W. M. (2001). Introduction. In Pitti, D. and Duff, W. M., editors, *Encoded Archival Description on the Internet*, pages 1–6. The Haworth Press, Inc.

Pitti, D. V. (1999). Encoded Archival Description. An Introduction and Overview. *D-Lib Magazine*, 5(11).

Pólya, G. (1937). Kombinatorische Anzahlbestimmungen fr Gruppen, Graphen und chemische Verbindungen. *Acta Mathematica*, 68:145–254.

Potter, M. (2004). *Set Theory and Its Philosophy*. Oxford University Press, Oxford, Oxfordshire, UK.

Prom, C. J. (2002). Does EAD Play Well with Other Metadata Standards? Searching and Retrieving EAD Using the OAI Protocols. *Journal of Archival Organization*, 1(3):51–72.

Prom, C. J. (2003). Reengineering Archival Access Through the OAI Protocols. *Library Hi Tech*, 21(2):199–209.

Prom, C. J. and Habing, T. G. (2002). Using the Open Archives Initiative Protocols with EAD. In Marchionini, G. and Hersch, W., editors, *Proc. 2nd ACM/IEEE Joint Conference on Digital Libraries, (JCDL 2002)*, pages 171–180. ACM Press, New York, USA.

Prom, C. J., Rishel, C. A., Schwartz, S. W., and Fox, K. J. (2007). A Unified Platform for Archival Description and Access. In Rasmussen, E. M., Larson, R. R., Toms, E., and Sugimoto, S., editors, *Proc. 7th ACM/IEEE Joint Conference on Digital Libraries, (JCDL 2007)*, pages 157–166. ACM Press, New York, USA.

Pumain, D. (2006). Introduction. In Pumain, D., editor, *Hierarchy in Natural and Social Sciences*, volume 3, pages 1–12. Springer, Berlin Heidelberg, Germany.

Ruth, J. E. (2001). The Development and Structure of the Encoded Archival Description Document Type Definition. In Pitti, D. and Duff, W. M., editors, *Encoded Archival Description on the Internet*, pages 27–59. The Haworth Press, Inc.

Sankoff, D. and Kruskal, J. B. (1983). *Time Wraps, String Edits, and Macromolecules*. Addison-Wesley, Reading, Massachusetts, USA.

Sattath, S. and Tversky, A. (1977). Additive Similarity Trees. *Psychometrika*, 42:319–345.

Shreeves, S. L., Kaczmarek, J. S., and Cole, T. W. (2003). Harvesting Cultural Heritage Metadata Using the OAI Protocol. *Library Hi Tech*, 21(2):159–169.

Silvello, G. (2008). Building a Distributed Digital Library System Enhancing the Role of Metadata. In *BCS-IRSG Symposium: Future Directions in Information Access*, pages 46–53. Published as part of the eWiC Series.

Simmhan, Y. L., Plale, B., and Gannon, D. (2005). A Survey of Data Provenance in e-Science. *SIGMOD Record*, 34:31–36.

Simon, H. A. (1962). The Architecture of Complexity. In *Proceedings of the American Philosophical Society*, volume 106, pages 467–482. American Philosophical Society.

Soergel, D. A. (2002). A Framework for Digital Library Research. *D-Lib Magazine*, 8(12).

Spath, H. (1980). *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*. Chichester: Ellis Horwood, West Sussex, England, UK.

Sugimoto, G. and van Dongen, W. (August 2009). Archival Digital Object Ingestion into Europeana (ESE-EAD Harmonization). Technical report, Europeana v1.0.

Tatarinov, I., Viglas, S., Beyer, K. S., Shanmugasundaram, J., Shekita, E. J., and Zhang, C. (2002). Storing and Querying Ordered XML Using a Relational Database System. In [Franklin et al., 2002], pages 204–215.

Tropashko, V. (2005). Nested Intervals Tree Encoding in SQL. *SIGMOD Record*, 34(2):47–52.

Tversky, A. (1977). Features of Similarity. *Psychological Review*, 84(4):327–352.

Van de Sompel, H. and Lagoze, C. (2007). Interoperability for the Discovery, Use, and Re-Use of Units of Scholarly Communication. *CTWatch Quarterly*, 3(3).

Van de Sompel, H., Lagoze, C., Nelson, M., and Warner, S. (2002a). Guidelines for Repository Implementers. Technical report, Open Archive Initiative.

Van de Sompel, H., Lagoze, C., Nelson, M., and Warner, S. (2002b). Implementation Guidelines for the Open Archive Initiative Protocol for Metadata Harvesting - Guidelines for Harvester Implementers. Technical report, Open Archive Initiative, p. 6.

Van de Sompel, H., Lagoze, C., Nelson, M., and Warner, S. (2003). The Open Archives Initiative Protocol for Metadata Harvesting (2nd ed.). Technical report, Open Archive Initiative, p. 24.

Vegas, J., Crestani, F., and de la Fuente, P. (2007). Context Representation for Web Search Results. *Journal of Information Science*, 33(1):77–94.

Vegas, J., de la Fuente, P., and Crestani, F. (2003). WebDocBall: A Graphical Visualization Tool for Web Search Results. In Sebastiani, F., editor, *Advances in Information Retrieval, 25th European Conference on IR Research, ECIR 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 351–362. Springer-Verlag, Heidelberg, Germany.

Verdier, N. (2006). Hierarchy: A Short History of a Word in Western Thought. In Pumain, D., editor, *Hierarchy in Natural and Social Sciences*, volume 3, pages 13–37. Springer, Berlin Heidelberg, Germany.

Vitali, S. (2010). Archival Information Systems in Italy and the National Archival Portal. In [Agosti et al., 2010c], pages 5–11.

VV. AA. (Version 5.2, 30/7/2010). Definition of the Europeana Data Model Elements. Technical Report of Europeana v1.0 in cooperation with EuropeanaConnect., Europeana v1.0.

Weibel, S. (1997). The Dublin Core: A Simple Content Description Model for Electronic Resources. *Bulletin of the American Society for Information Science and Technology*, 24(1):9–11.

Weikum, G., König, A. C., and Deßloch, S., editors (2004). *Proceedings of the ACM SIGMOD International Conference on Management of Data, (SIGMOD 2004)*. ACM Press, New York, USA.

Wojcik, R. H., Harrison, P., and Bremer, J. (1993). Using Bracketed Parses to Evaluate a Grammar Checking Application. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, ACL '93, pages 38–45, Morristown, NJ, USA. Association for Computational Linguistics.

Xu, L., Ling, T. W., Wu, H., and Bao, Z. (2009). DDE: From Dewey to a Fully Dynamic XML Labeling Scheme. In Çetintemel, U., Zdonik, S. B., Kossmann, D., and Tatbul, N., editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2009*, pages 719–730. ACM Press, New York, USA.

Zezula, P., Amato, G., Dohnal, V., and Batko, M. (2006). *Similarity Search. The Metric Space Approach*. Springer, Berlin Heidelberg, Germany.

Acknowledgements

There were many people who supported me in the development of the work that I presented in this thesis. I am grateful to the people of the Department of Information Engineering of the University of Padua, in particular to all the members of the Information Management System research group.

I am really pleased to give my sincere thanks to the people I met at the School of Informatics of the University of Edinburgh where I spent seven enriching months. A special thanks goes to Peter Buneman who welcomed me in Edinburgh and worked with me to several aspects related to this work. I want to thanks also Floris Geerts and James Cheney for many useful discussions.

A special thanks goes to the people who collaborate with me in the SIAR project: Massimo Canella and Andreina Rigon of the "Regione del Veneto", and Erilde Terenzoni and Cristina Tommasi of the "Sopraintendenza archivistica per il Veneto". To this purpose I want to thanks Giorgetta Bonfiglio-Dosio of the History Department of the University of Padua who collaborated with us in the first part of the SIAR project.

I want to thanks Max Dekkers for the talk we had after the PROMISE kick-off meeting in Padua about Dublin Core, EAD and ontologies. I thanks Leif Azzopardi and Stephen Robertson for their useful suggestions during FDIA 2008 in London.

Lastly, I want to thanks the next three great thinkers who filled me with awe.

"Intuition and concepts constitute [...] *the elements of all our knowledge, so that neither concepts without an intuition in some way corresponding to them, nor intuition without concepts, can yield knowledge."*

Immanuel Kant

"Science is what we understand well enough to explain to a computer. Art is everything else we do."

Donald Knuth

"I ran into Isosceles. He had a great idea for a new triangle!"

Woody Allen