

Analogical Similarity of Objects: A Conceptual Modeling Approach

Doctoral Dissertation

George E. Spanoudakis

Department of Computer Science University of Crete

Abstract

This dissertation describes a computational model for detecting analogies between objects and estimating their similarity, based on an analysis of conceptual descriptions of these objects.

The proposed model adopts a representation framework with three general-purpose semantic modeling abstractions (i.e. classification, generalization and attribution), which are adequate for building conceptual models for a wide spectrum of application domains and tasks. Furthermore, their semantics inherently reveal analogies between such conceptual models.

The detection(elaboration) of analogies is achieved by using metrics measuring the distance between conceptual descriptions of objects with respect to each of the prescribed abstractions. The similarity model we propose, consists of three such metrics, namely the classification, generalization and attribution distances.

These metrics distinguish the importance of the specific relations they take into account. In particular, the attribution distance is informed about the importance of the attributes of the involved objects by the so called *measures of salience*. These measures are estimated from beliefs about three properties of attributes (i.e. the *charactericity*, *abstractness* and *determinance*), which are introduced by our model as predicting the importance of attributes to the elaboration of analogies. The beliefs about the truthness of these properties are measured by functions taking into account specific patterns of representing attributes in conceptual models.

The exact criteria for hypothesizing analogies in the first place and the functions measuring distances and salience ensure the domain independency of the proposed model. Also, they make it operational under non uniform representations of objects(i.e. representations which are not based on specific sets of relations) and tolerant to the absence of explicitly asserted knowledge determining important factors for the detection of analogies. Hence, the proposed similarity model overcomes identified problems of other models for elaborating analogies[Ked88,Hall89] and offers a generally applicable computational mechanism for this elaboration.

The model has been implemented and integrated with the *Semantic Index System*(i.e. a system for representing scientific knowledge and engineering artifacts). The resulted prototype (SIS/SA) was used in a couple of tasks in the domain of software

engineering, namely the analogical reuse of requirements specifications and the integration of requirements viewpoints. These applications demonstrated the applicability of the proposed model in complex tasks involving interdomain analogical reasoning.

SIS/SA was also used in experiments evaluating the consistency of similarity estimates generated by our model with human assessments, as well as, the recall and the time performance of it. Albeit preliminary, these experiments indicated a promising behaviour of the similarity model, with respect to the mentioned aspects of evaluation.

Chapters

Chapter 1: Overview of Analogical Similarity

Chapter 2: Elaboration of Analogies

Chapter 3: Distance and Similarity Functions

Chapter 4: Saliency Functions

Chapter 5: Implementation and Evaluation of the Model

Chapter 6: Illustrative Applications

Chapter 7: Comparison with General Models of Analogical Reasoning

Chapter 8: Conclusions and Future Research

Glossary(English-Greek)

Bibliography

Chapter 1

Overview of Analogical Similarity

The remarkable ability of humans to understand novel situations by analogy to familiar ones and, to solve new problems by remembering solutions to already solved analogous problems, has motivated the study of analogical reasoning as a non deductive paradigm of reasoning.

Inference by analogy attempts to derive conjectures about an item B (referred to as the *target* analog [Hall89]), given that it is analogous to another item A (referred to as the *source* or *base* analog[Hall89]). Such conjectures comprise known aspects of A, which are not known yet about B. These aspects may be properties of B, predictions about its behavior or solutions to problems related to it[Niin88].

Analogs may be single individuals or complete systems of natural or nominal kind objects. For instance, analogies may exist between different kinds of animals, different engineering artifacts(e.g. a water pipe and an electric cable[Win80]) or different social structures and institutions (e.g. analogies between business and political structures). Analogies may also exist between classes of phenomena or concepts(e.g. religions may be thought of as being analogous to political ideologies).

As pointed in[Niin88], analogical reasoning is an indispensable form of scientific reasoning, which may complement other forms of reasoning(e.g. deduction) in cases where:

- they cannot be carried out due to the lack of available resources, such as computational time or memory;
- experimentation with target analogs is prohibited due to moral or legal reasons(e.g. experimentation with animals in medicine gives initial evidence about the effectiveness of drugs justified by biological analogies between these animals and humans);
- knowledge about target analogs is incomplete due to practical reasons or technological limitations and thus, insufficient for drawing the required conjectures about them(e.g. analogical conjectures about the possibility of life on other planets of the solar system, in the previous century)

In view of such situations, analogical reasoning has been applied to or suggested as a plausible reasoning paradigm for a variety of application domains and tasks, such as:

- law(e.g. reasoning about analogous legal cases[Ash90]);
- medicine(e.g. diagnosis in radiology[FG91]);
- economics(e.g. foreign trade negotiations[CPS91], quality assurance of products[Allen94]);
- politics(e.g. dispute mediations[Sim85]);
- mechanical engineering(e.g. design of hydro-mechanical systems[SN91]); and,
- software engineering(e.g. reuse of requirements specifications[MH91,MS92] or other software artifacts[SC94a,SC94b], logic programming[TB91]).

Analogy and analogical reasoning have been studied in cognitive science, psychology, philosophy and artificial intelligence. Research in these disciplines focuses on different aspects of analogy. Cognitive scientists and psychologists are primarily concerned with how humans remember analogs and reason by analogy[GL85,GT86,Nov88,SH88]. Philosophers concentrate on prerequisite conditions for drawing valid analogical inferences[Aga88,Kuip88,Niin88]. Finally, AI researchers focus on the development of computational models and systems for reasoning by analogy(see references in our survey of such models in chapter 2 and in other relevant surveys[Hall89, Ked88]).

1.1 Computational Study of Analogical Reasoning: An Open Challenge

As Paul Thagard points out in [Thag88], studying analogy with a focus on its computational aspects is important in its own right. Computational ideas and implementations make possible the development of detailed models of mental structures and processes related to analogical reasoning, predicted by research in other disciplines. Developed computational models can be used to postulate and judge these predictions, in comparison with human performance. In addition, the development of computational models of analogical reasoning is justified by the possibility to achieve efficient reasoning, applicable in complex tasks, which can hardly be dealt with in due time or as exhaustively as they should be, by humans.

Although some authors insist that computational models of analogical reasoning must be models of human cognition [Gen83, Nov88, Gen88a], there is no reason why all should be. Indeed, research in disciplines like cognitive science and psychology does not always provide unique and clear answers about the underlying structures and processes of analogical reasoning [Hall89]. Their findings may be partial or even mutually conflicting (e.g. the empirical justification of both the syntactic and the determination approach to analogical inference [GT86, HK87b, Nov88], described later in this chapter).

Thus, without neglecting the possibility of building computational models on the basis of empirical and theoretical developments of such disciplines, we claim that the computational study of analogy can also make its own contributions to the subject matter. After all computational models might eventually perform analogical reasoning better than humans, subject to a correct exploitation of the very strengths and computational abilities of computers over human brains [Thag88].

In our opinion, computational models intended to support analogical reasoning without being restricted by the kinds of the involved analogs and the purpose of reasoning (i.e. the particular conjectures being sought), must satisfy the following requirements:

- They should detect analogies in a domain-independent way so as to be applicable in a large variety of tasks, especially those involving analogs from different domains.
- They should be robust to different structural configurations of analogous elements in descriptions of analogs.

- They should be operational under non uniform representations of analogs(i.e. representations using different sets of features).
- They should distinguish elements in descriptions of analogs having a particular importance to analogy(usually referred to as *salient* elements) in order to differentiate between true analogies and superficial resemblances.
- They should not rely on extensive amounts of knowledge, or else they would be inevitably bounded by the knowledge acquisition bottleneck.
- They should require as little information from users as possible, or else they might be impractical.

The rather dense research effort regarding the computational aspects of analogical reasoning has not delivered models satisfying all these requirements to the present day. In fact, proposed computational models tend to be:

- overly sensitive to the representation, the kinds and the domains of analogs involved or,
- highly dependent to the specific purpose of analogical reasoning;
- designed to operate using extensive amounts of knowledge determining critical aspects for the detection of analogies; or,
- operational only when their users are able to provide them with various kinds of information relevant to the detection of analogies (e.g. user-supplied distance measures).

This failure, which in turn is argued on the basis of specific solutions offered by various models in chapter 2, justifies the research focus of this thesis.

We concentrate on the computational aspects of analogical reasoning, investigate the assumptions and the implications of solutions provided by existing models and develop a computational model for identifying analogies between arbitrary kinds of analogs(referred to as the elaboration of analogies in the literature[Hall89]), that satisfies to a large extent the outlined requirements.

In the following, we introduce this model in reference to the prominent approaches to analogical reasoning, that have appeared in the computational[Rus88] and the cognitive science[Nov88] literatures.

1.2 Main Approaches to Analogical Reasoning

Inference by analogy can be formally described by the following rule[Niin88]:

IF A, B are *analogous* with respect to

f_1, f_2, \dots, f_n

and $g(A)$ THEN

infer $g(B)$

where

- A, B are the source and the target analogs, respectively;
- f_1, f_2, \dots, f_n are their analogous characteristics; and,
- g is a characteristic of A , which is not originally known about B , but is inferred about it on the basis of the other analogous characteristics.

Different interpretations of the constituents (and their interrelations) of this rule characterize the different theoretical approaches to analogical reasoning. There exist three different approaches[Rus88]:

- the *determination* approach ;
- the *syntactic similarity* approach; and,
- the *probabilistic similarity* approach.

The Determination Approach. This approach requires the existence of knowledge determining the elements of two analogs which are important to analogies. This knowledge expresses causal dependencies between characteristics f_i of analogs and specific conjectures g about them. Determinant (or causal) knowledge guides reasoning to focus only on relevant characteristics of analogs and ignore the other.

As it is evident from computational models adopting it (see models of analogical problem solving in chapter 2), the determination approach is powerful in contexts where reasoning is carried out for achieving some well-defined purpose (e.g. solving specific problems). This purpose determines the conjectures precisely. Thus, it is easier to identify the characteristics of analogs, which are causally related to these conjectures. The determination approach is appropriate for domains characterized by well-understood causalities.

Due to the focus on goal attainment, this approach is oftenly referred to as the *pragmatic approach* [Nov88].

The Syntactic Similarity Approach. This approach realizes that knowledge determining factors causal to analogies, does not always exist or cannot be readily acquired and expressed. Consequently, it advocates syntactic criteria for distinguishing the characteristics f_i of analogs that should be taken into account in trying to detect analogies and draw inferences from them.

In essence this approach is heuristic[Rus88]. Conjectures are justified by the ability of the employed criteria to distinguish characteristics really important to analogical inference. Nevertheless, such conjectures must also be verified empirically.

Only a few of the proposed computational models adopt the syntactic approach. These models, as discussed in chapter 2, are not particularly powerful in discriminating important characteristics. Furthermore, they adopt criteria for detecting analogies, which are overly sensitive to the configurations of analogs or they can only be approximated computationally. Thus, the computational realizations of this approach may prove impractical in many applications.

The Probabilistic Similarity Approach. This approach is also introduced to overcome the absence of causal knowledge[Rus88]. It is based on the intuition that, even in situations where a system is extremely ignorant of causal factors, such factors are likely to exist in descriptions of analogs. In other words, some of the characteristics of analogs will be relevant to the conjectures being sought.

In such cases, a large similarity, measured by the number of the shared characteristics of two analogs, serves to increase the likelihood that the causally relevant characteristics, albeit unknown, will be included in the commonality. Hence, it is plausible to derive conjectures based on a high similarity between analogs. Some authors use measures of similarity to derive probabilities about the validity of inference[Rus88, Niin88, Kuip88].

The main weakness of the probabilistic similarity approach lies to its strong assumption about the nature of the representations of analogs. According to it, each characteristic of an analog corresponds to a causal factor. This hypothesis has been questioned on the basis of empirical evidence[Gen83, Ross88, Nov88]. Consequently, the heuristic validation of conjectures drawn on the basis of this approach could be neglected.

To our knowledge, none of the proposed computational models of analogical reasoning adopts the probabilistic similarity approach.

1.3 Proposal: A Conceptual Modeling Approach

In this thesis, we claim that a computational model must adopt the syntactic similarity approach, so as to have a potential to fulfill the requirements of section 1.1.

We feel that, in spite of the weaknesses of the models adopting it, the syntactic similarity approach has the following significant advantages over the other two:

- it is domain independent;
- it is not bounded by the acquisition of causal knowledge; and,
- it can support reasoning in contexts lacking a precise purpose of reasoning.

On condition of employing criteria which can discriminate between important and unimportant characteristics of analogs and, which are not overly sensitive to the exact configurations of analogs a syntactic similarity model is potentially applicable to a large variety of tasks, amenable to analogical reasoning.

The model proposed by this thesis, adopts the syntactic similarity approach. Its main underlying assumption is that the effectiveness of this approach can be improved by selecting a set conceptual modeling abstractions for describing analogs, whose semantics inherently reveal analogies. Then analogies can be detected using criteria based on these semantics.

In particular, we have selected three such modeling abstractions, namely *classification*, *generalization* and *attribution* [BMW86, HK87a, PM88, KMSB89, MBJK90]. Classification allows the grouping of entities, with common features into classes abstracting these features. Generalization allows the extraction of common features from two or more classes into a common general class, which suppresses their detailed differences. Finally, attribution allows the association of properties and relations with individuals and/or classes.

These abstractions constitute a representation framework which, due to its high expressive power, well-defined semantics and closeness to the human way of thinking about the construction of complex models [FK85], is adequate for constructing *conceptual*

models(i.e. human-oriented descriptions of artifacts), in many different application domains.

Furthermore, the very semantics of the selected modeling abstractions enable the definition of criteria for:

- deciding whether or not elements in conceptual models of analogs are analogous to each other, and
- discriminating between such elements in terms of their importance to the detection of analogies.

Based on conceptual models of analogs built according to these abstractions, we measure conceptual distances. We introduce three distance metrics, namely the *classification*, the *generalization* and *attribution* distances, which estimate the distances between conceptual models of analogs with respect to each of the prescribed abstractions.

The distance over classification indicates whether or not two objects have an analogous substance. The distance over generalization gives a coarse-grain indication of the semantic differences of objects. Finally, the distance over attribution establishes a mapping between the analogous attributes of objects and estimates a total distance measure for this mapping.

The partial measures obtained by these metrics are aggregated in an overall distance measure, which is transformed into a similarity measure indicating the aptness of the detected analogies. Partial measures are aggregated to deal with cases, where conceptual models of analogs are incomplete with respect to any of the prescribed abstractions.

None of the metrics uses any sort of distance measures either a-priori specified or explicitly provided by the user, like other computational models do. This choice overcomes problems that may arise, when such measures are not readily available.

Our model distinguishes between attributes of different importance to the detection of analogies. Importance is quantified by a measure called *salience*. Salience is estimated as belief that an attribute has three properties, which indicate its importance to analogy, namely *charactericity*, *abstractness* and *determinance*.

These properties are logically connected with specific patterns of representing attributes. Then, belief to their validity is obtained by belief functions(in the sense of the

Dempster-Shafer Theory of Evidence[Sha75]), which measure the extent to which such modeling patterns occur. In this way, it is not necessary to assume any specific representation primitives for distinguishing attributes important to analogies or any explicitly provided measures for quantifying importance. However, as discussed in chapter 4, our model can accommodate beliefs provided by the user about the prescribed properties of attributes, subject to certain consistency conditions. This possibility is explored for cases where it would be possible or necessary to improve the accuracy of the default distinctions made by the model.

In summary, our model of similarity introduces a set of functions, whose definitions upon only the prescribed general semantic modeling abstractions, make it applicable in a wide spectrum of problems and application domains.

The potential of our model as a computational model of elaborating analogies capable of supporting non trivial tasks is demonstrated by applying it to a couple of tasks(i.e. the analogical reuse of requirements specifications [MH91a,MH91b,MS92,SC94a] and the integration of requirement-viewpoints[LF91]) in the domain of software engineering.

It must be pointed out that the selected abstractions are not the only discussed in the literature. Other semantic modeling abstractions (e.g. *association* and *aggregation*) have been also proposed[PM88]. Our selection is due to the coupling between the semantics of classification, generalization and attribution and the notion of analogy. This selection does not preclude future extensions of our model so as to cover a wider framework including those additional abstractions.

We characterize our approach to the elaboration of analogies as a conceptual modeling approach[SC93], since all computations involve different types of analysis of conceptual models of analogs.

1.4 Main Contributions

The main contributions of this doctoral research are summarized into the following:

- It formally defines and implements a general and domain independent computational model of elaborating analogies. This model does not require the acquisition of specific forms of causal knowledge for determining analogies and is not sensitive to exact configurations of analogs. It distinguishes attributes in conceptual descriptions of analogs

having a different significance to analogies and quantifies this significance, without being explicitly informed about it by the user. Also, it can detect analogies in polynomial time, even if implemented sequentially. Due to these characteristics, it outperforms other general computational models of elaborating analogies, which lack one or more of these characteristics.

- It verifies the potential of the syntactic approach as an effective computational approach to elaborating analogies. In particular, it shows that it is possible to detect analogies from representations of analogs built according to three general purpose conceptual modeling abstractions (i.e. classification, generalization and attribution) based only on criteria exploiting the semantics of these abstractions. In this way, it also illustrates the role of conceptual modeling in non deductive forms of reasoning, like reasoning by analogy.
- It applies the developed model to complex tasks in the area of software engineering, in particular to reuse and integration of requirements specifications. This application shows that the developed model for elaborating analogies is operational in spite of the representation diversity, incompleteness and complexity of the involved artifacts. It also shows that the model can be used to detect analogies between artifacts from different domains and is not bounded by the knowledge acquisition bottleneck. Thus, it meets important pragmatic requirements of applying analogical reasoning to the selected tasks, unlike other models proposed for the same purpose.
- It isolates four basic computational aspects of elaborating analogies (these are the representation of analogs, the mapping criteria, the discrimination of salience and the measurement of the aptness of an analogy) and points out how different solutions to these aspects depend on each other.

1.5 Thesis Organization

The remaining seven chapters of this thesis present the results of our research into the computational aspects of elaborating analogies.

Chapter 2 identifies the basic computational aspects in the elaboration of analogies and reviews models proposed in the literature. It points out the relation between the purpose of reasoning these models are meant to serve and the solutions they adopt. It discusses the merits of these models in supporting analogical reasoning in general (i.e. without

assumptions about particular characteristics of application contexts and tasks). Finally, it introduces our conceptual modeling approach to overcome identified limitations of the reviewed models.

Chapter 3 axiomatically defines the representation framework of the similarity model. This axiomatization provides the basis for formally defining its distance functions. The properties of each function are discussed and their role in the elaboration of analogies is illustrated using specific examples.

Chapter 4 elaborates on the problem of distinguishing between attributes that have a different importance to the detection of analogies. Importance is realized as a compound property of attributes, called *dominance* of attributes. Dominance is formally defined on the basis of three other primitive properties, namely *charactericity*, *abstractness* and *determinance*. These properties are defined by conditions regarding the modeling of attributes, which are consequently used to derive functions measuring the belief about their satisfiability. The defined functions are formally interpreted as belief-functions in the sense of the Dempster-Shafer theory of evidence[Sha75]. In the same chapter, we also investigate into the possibility of informing the model externally with further beliefs, so as to revise its default measures of salience.

In chapter 5, we develop the algorithms computing the distance and the salience functions for similarity analysis. The time and space complexities of these algorithms are analyzed. That chapter also describes a prototype implementation of the algorithms, which is used for a preliminary empirical evaluation of the model regarding:

- the consistency of the similarity estimates produced by the model with relevant human estimates;
- the recall performance of similarity model, in analogical retrieval; and,
- the time performance of similarity model.

The results of this evaluation are presented and discussed.

Chapter 6 describes the application of similarity analysis in two tasks amenable to analogical reasoning in the domain of software engineering: reuse and integration of requirements specifications. The complexity and other characteristics of the selected domain and the tasks in it are highlighted so as to demonstrate the potential of similarity analysis

as a model applicable to complex real applications involving reasoning by analogy. In the same chapter, we also discuss the advantages of our model by comparison to models of analogical reasoning developed to support exactly the selected tasks.

Chapter 7 compares the similarity model with other general computational models of analogical reasoning. Comparisons are made in reference to a set of criteria regarding the flexibility, the computational complexity, the domain independence, the ability to discriminate salience, the sensitivity to causal knowledge and the required user interaction of each model. These comparisons indicate why the similarity model outperforms other models in elaborating analogies.

Finally, chapter 8 summarizes the main results of this research, discusses aspects that should be further investigated and points out open research issues, which although outside the original scope of this research, have emerged along the way.

Chapter 2

Elaboration of Analogies

2.1 Introduction

Analogical reasoning has been described by an abstract process model including four stages[Hall89, Ked88]:

- (1) *recognition* (or retrieval) of a candidate source given a target analog;
- (2) *elaboration* of a mapping between the analogous characteristics of two analogs;
- (3) *evaluation* (or justification) of the conjectures drawn about the target analog; and,
- (4) *consolidation* of detected analogies into structures making possible their reference in other contexts.

In this thesis, we focus on the computational aspects of the elaboration of analogies. Elaboration is crucial since it provides the basis for the application of the rule of analogical inference (see chapter 1). Thus, it receives the largest share of attention, among the four stages of analogical reasoning in existing computational models[Hall89].

In the following, we review computational models of analogical reasoning, discuss their merit as domain independent models and the properties that should be owned by such

models. Finally, we introduce the similarity model.

2.2 Basic Aspects of Elaboration

Elaboration in analogical reasoning addresses a central problem: the identification of the elements of a source and a target which are important for detecting their analogies and the construction of a mapping between them, reflecting these analogies, as precisely as possible.

Computationally, elaboration must address four aspects:

- (1) the representation of analogs ;
- (2) the criteria for mapping elements of analogs ;
- (3) the distinction of salience of different elements in descriptions of analogs; and,
- (4) the estimation of the aptness of an analogy.

Representation. Representation deals with the modeling primitives, which are necessary for describing analogs, so as to make possible the detection of analogies between them. Modeling primitives must be *symbolically sufficient*[Win80]. Symbolic sufficiency implies that the primitives offered in a representation framework must allow the identification of the elements of two analogs, which can be placed in correspondence and which are also important for the detection of analogies[Thag88].

A representation framework must be expressive enough to allow the description of analogs from different domains. Otherwise, it would not be a good candidate for domain independent analogical reasoners.

Representation frameworks devised solely to accomplish analogical reasoning (e.g. problem-oriented representations) are only useful for autonomous systems of analogical reasoning. Such frameworks may considerably limit the ability to perform analogical reasoning on descriptions devised so as to be amenable to other forms of computational reasoning, too. In such cases, the combination of analogical reasoning with other forms of reasoning, which is often necessary for dealing with fairly complex problems in real application domains[Allen94], is difficult.

Mapping Criteria. The mapping criteria determine which elements of two analogs can be placed in correspondence and which set of such correspondences reflect an analogy, as

precisely as possible. Mapping criteria depend on the scope, the purpose and the representation adopted by a system of analogical reasoning.

Mapping criteria can be divided into:

1. *Cardinality criteria*, which constrain the cardinality of the mapping between the elements of two analogs (e.g. 1-1 mappings).
2. *Type compatibility criteria*, which determine the conditions under which distinct elements of analogs can be mapped onto each other (e.g. identity based mapping).
3. *Structure preserving criteria*, which constrain mappings so as to preserve the configurations of elements in both the involved analogs (the *systematicity principle* in [Gen83]).
4. *Knowledge preserving criteria*, which restrict mappings so as to include only elements of analogs, important to analogies, indicated by special knowledge (e.g. abstraction based mappings in [Grein88a]).
5. *Consistency preserving criteria*, which restrict mappings so that the mapped elements be consistent with existing knowledge about the target analog.

Cardinality and type compatibility criteria are less strict than structure, knowledge and consistency preserving criteria. They determine an overall search space of valid mappings but cannot lead to a single selection from this space. Thus, they could be referred to as *validity criteria*.

On the other hand, structure, knowledge and consistency preserving criteria introduce more strict conditions, which usually enable the selection of a single mapping from the search space of valid mappings. Thus, they could be referred to as *preference criteria*.

Saliency. The computation of saliency addresses the fundamental question: which elements of analogs are important for analogies and thus should influence more their elaboration. The computation of saliency, like the mapping criteria depends on the scope, the purpose and the representation framework of analogical reasoning. Many systems introduce distinctions of saliency using knowledge preserving criteria, employed in mapping. Other systems attempt to derive saliency from syntactic aspects of the representation of elements.

Aptness. Aptness refers to qualitative or quantitative assessments about the strength of an analogy. The estimation of aptness serves many purposes. During the elaboration, it may be used as a criterion for selecting among different valid mappings between analogs. At the same stage, it can be also used for selecting the most promising among different candidate analogs. Finally, it may be used as a confidence measure about the ability to derive conjectures from elaborated analogies.

2.3 A Survey of Computational Models of Analogical Reasoning

In the following, we review the solutions given to each of the four aspects of elaboration by proposed models of analogical reasoning. These models can be organized into the taxonomy, presented in figure 2.1, according to the purpose they serve and the domains of the analogs involved.

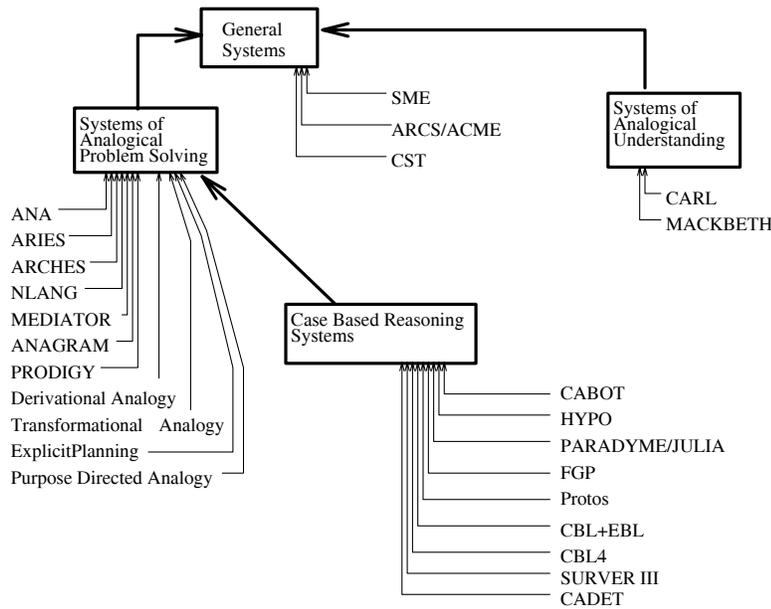


Figure 2.1: A Taxonomy of Models of Analogical Reasoning

This taxonomy distinguishes four classes of models, namely:

- (1) models of general analogical reasoning, in which the detection of analogies may have any possible purpose and the analogs may belong to different or the same domains;

(2) models of analogical understanding, in which analogies are detected for explaining unfamiliar domains in terms of familiar ones;

(3) models of analogical problem solving, which detect analogies so as to solve specific problems and involve analogs that may belong to different or the same domain; and,

(4) models of case-based reasoning, a special kind of models of analogical problem solving, which detect analogies between items in the same application domain.

Other surveys of models for analogical reasoning, presented in the literature, classify models using either chronological criteria or a combination of chronological criteria with criteria about the purpose of reasoning they perform[Ked88,Hall89].

2.3.1 General Models

We have found three general models of analogical reasoning in the literature. These include the *Structure Mapping Theory(SMT)*[Gen83], implemented by the *Structure Mapping Engine(SME)*[FFG90], the *Constraint Satisfaction Theory*[Thag88], implemented by two systems called ACME [HT89] and ARCS[THNG90] and, the theory of *Constrained Semantic Transference(CST)*[Ind86], computationally approximated by the *Approximate Semantic Transference*[Ind85].

These models have been introduced as realizations of theories on how humans reason by analogy. SME focuses on elaboration, CST focuses on elaboration and justification and the Constraint Satisfaction Theory focuses on retrieval and elaboration, respectively.

Representation. As general models, SME, ARCS, ACME and AST adopt domain independent representation frameworks based on variations of predicate calculus.

In particular, AST uses a typed predicate calculus. SME also uses a typed predicate calculus and distinguishes predicates expressing individual or constants, attributes, functions and relations. Attribute-predicates are used to express properties of individuals, function-predicates are used to express correspondences between individuals and relation-predicates are used to express relations of arbitrary arity between them. Analogs are represented as aggregations of entities, attributes, functions and relations.

ARCS and ACME combine predicate calculus with a frame-based representation. Analogs are represented as aggregations of predicates. Predicates are associated with frames in a-priori existing taxonomies. These frames represent general concepts, modeled after a

lexical reference system, the *WorldNet*[MFKM88], and are used to disambiguate the semantics of their associated predicates. Frame taxonomies are organized by means of *kind-of*, *part-of*, *synonymy* and *antonymy* relations[THING90]. Furthermore, ARCS and ACME offer special relations, expressing knowledge about the pragmatic utility of particular predicates to specific purposes of reasoning.

Mapping Criteria. SME adopts cardinality, type compatibility and structure preserving criteria, but not knowledge and consistency preserving ones. It allows only isomorphic mappings between identical predicates which also do not violate the relational structure of the involved analogs (i.e. the *systematicity principle*[Gen83,Gen88a]). The relational structure is preserved in a top-down fashion. When two relations are mapped onto each other, their corresponding arguments must be mapped onto each other, too. These criteria preserve the domain independence of the model, since they do not utilize any special forms of knowledge about the source or the target domains.

ACME on the other hand adopts cardinality, type compatibility, structure and knowledge preserving criteria. It allows N:1 mappings and relies on the same structure preserving criteria as SME. It also allows mappings between non identical predicates, as long as they have the same arity and the frames of their associated concepts are connected by some of the provided taxonomic relations.

ACME tends to select mappings between predicates characterized as important either by some external cooperating tutor or due to their involvement in special, importance indicating relations(i.e. goals and state conditions in problem-oriented representations of analogs).

It must be pointed out that none of the employed criteria is enforced in a strict logical sense by ACME. Their satisfaction is subject to the operation of a connectionist network[Gro78]. The units and the links of this network express the mapping hypotheses and the pairwise supports or exclusions between them, respectively. The eventual mapping depends on the activation levels of the network units, which are initially set to standard quantities and then they are iteratively updated according to the equation for updating activations suggested by Grossberg[Gro78].

AST also allows N:1 mappings. Due to its *admissibility* principle[Ind86], constants and individuals from different domains cannot be mapped, unless they have the same type.

Also, predicates do not map unless they are identical. More importantly however, AST accepts only the, so called, *coherent* mappings. Coherence is a consistency preserving criterion. According to it, individuals cannot be mapped if they are involved in predicates, whose conjecture about the target analog would be inconsistent with existing knowledge about it. Consistency of first-order formulas is an undecidable problem in general. Thus, the coherency criterion is only approximately checked(i.e. it is checked within limited computational time).

Salience of Elements. SME distinguishes between predicates of different salience according to their arity and order. Predicates over a single argument, that correspond to attributes and functions of analogs, are not considered important to analogies and consequently they are not taken into account in elaboration. On the other hand, predicates over two or more arguments, as well as predicates having as arguments other predicates are considered to represent elements of analogs important to the detection of analogies and thus become the focal points in elaboration. These syntactic distinctions are cognitively grounded[Gen88b].

ACME and ARCS provide an implicit computational account of salience. They favour mappings between predicates which are characterized as important either by an external tutor or by their involvement in specific relations known to indicate this importance.

AST does not distinguish between elements of different salience.

Aptness of Analogy. All three general models of analogical reasoning measure the aptness of detected analogies.

SME provides a measure called *structural evaluation score*[FFG90]. This score is obtained as the sum of evidence measures about the importance of pairwise mappings that constitute isomorphisms between analogs. These evidence measures depend on the types of the mapped predicates(i.e. they are a-priori defined for each such type). Proceeding in a top-down sequence, by first mapping predicates and then their arguments, SME assigns a fraction of the evidence, that is assigned to already mapped predicates, to the mappings of their arguments. Salience measures are used for assessing the strength of detected analogies, and selecting both among alternative mappings and different candidate analogs.

ACME and ARCS also offer a measure of aptness, which results from the operation of their connectionist network.

Finally, AST computes aptness as the ratio of the number of the consistent conjectures that can be drawn for the target analog, given its mapping to the source. Aptness in CST provides a measure for evaluating analogies rather than elaborating them.

2.3.2 Models of Analogical Understanding

The models of analogical understanding elaborate analogies so as to promote the comprehension of a less familiar domain(i.e. the *target domain*) in terms of a more familiar one(i.e. the *source domain*). Analogies are used for answering questions about the target domain.

Although classified in different ways in other surveys [Hall89,Ked88], two models: a model developed by Winston[Win80](referred to as MACKBETH in[Hall89]) and CARL[Bur86], could be classified as models of analogical understanding, according to our point of view.

Representation. Like general models, models of analogical understanding adopt general representation frameworks. In particular, MACKBETH uses semantic networks and CARL adopts a frame-based representation allowing also the organization of frames into semantic taxonomies. In addition, CARL realizes special relations that can be used to assert elements, which are causally important to particular analogies.

Mapping Criteria. Both the reviewed models adopt knowledge preserving criteria for elaborating analogies. Causal knowledge is given by an external tutor. This type of acquisition of causal knowledge indicates the difficulty of eliciting it when the purpose of reasoning is as loose and general as understanding.

Furthermore, MACKBETH and CARL do not map relations of distinct analogs unless they have the same arity and their corresponding arguments belong to the same classes. In addition, CARL maps only predicates of the same class.

MACKBETH and CARL differ regarding their structure preserving and cardinality criteria. MACKBETH allows only isomorphic mappings. Unlike it, CARL allows N:1 mappings and also adopts the top-down structure preserving criterion.

Salience of Elements. Both the reviewed models realize as salient only the elements of the involved analogs, which are either assessed as important by the external tutor or participate in special relations known to indicate importance (e.g. integrity constraints [Win80]).

Aptness of Analogy. MACKBETH measures aptness. It uses a similarity metric which is based on the *contrast model of similarity*, proposed by Tversky [Tve77] (i.e. a symmetric linear combination of the numbers of the mapped and the unmapped elements of two analogs). MACKBETH exploits aptness measures in order to evaluate the quality of detected analogies and to select a single source analog among competing ones.

CARL on the other hand, neglects the need for measuring aptness. Aptness measures are useless to it, since instead of choosing a single best source analog, given a target, CARL tries to combine analogies detected between all the candidate sources and the target. Furthermore, evaluation of analogies in CARL, is based on an assessment of the external tutor and thus, it does not have to take aptness measures into account.

2.3.3 Models of Analogical Problem Solving

The models reviewed in this section concentrate on problem solving by analogy. This pragmatic purpose distinguishes them from other models of analogical reasoning.

Models of analogical problem solving include ANA [McD79], explicit planning [Mun81], MEDIATOR [KSS85], ARCHES [BC85], purpose-directed analogy [Ked85], transformational and derivational analogy [Car83, Car86, Mos89], NLANG [Grein88a, Grein88b], PRODIGY [VC91a, VC91b] and ANAGRAM [Cook91]. The basic computational solutions with respect to representation, mapping, salience and aptness offered by these models are summarized in the following.

Representation. Representations in models of analogical problem solving are always problem-oriented. They comprise sets of initial problem states or conditions related with goals to be achieved. States, conditions and goals act as types for predicates or slots of frame structures, which are used for describing analogs.

This kernel representation scheme is adopted by PRODIGY [VC91a, VC91b] and ARIES (i.e. the system implementing the model of transformational analogy [Car83]). Other models extend it with further types of predicates, specific to the problems they

address or solution techniques they employ. For example, the derivational analogy model[Car86], which focuses on detecting analogies between problem solving episodes, provides special constructs expressing: operators applicable to each problem state, arguments about available choices, and reasons for successes and/or failures.

A very important feature of representations for analogical problem solving is the obligation to connect elements in the descriptions of analogs using relations that express specific causal dependencies (e.g. decompositions of goals into subgoals). In particular, NLANG realizes such causal dependencies through its *generalization abstractions* [Grein88a,Grein88b], MEDIATOR through its *generalized problem solving episodes*[KSS85], ARIES through its *invariance hierarchy*[Car83], derivational analogy model through its *derivational traces*[Car86], ARCHES through its *dependency relations*[BC85] and the purpose-directed analogy model through its *causal relations*[Ked85,Ked88].

Mapping Criteria. Elaboration for analogical problem solving is based on all but the consistency preserving criteria of mapping.

Usually only isomorphic mappings are allowed. This restriction is adopted by all models except ARCHES, which allows N:1 mappings.

Type compatibility is normally decided on the basis of the identity of the elements to be mapped(i.e. predicates or frame-slots). Some models(e.g. PRODIGY) adopt less strict compatibility criteria requiring that mappable elements must be only of the same kind(e.g. goals, states, conditions) but not necessarily identical.

However, the distinguishing mapping criteria of models of analogical problem solving are the knowledge preserving ones. Elements in the descriptions of analogs are not considered for mapping, unless they participate in causal structures determining what is important for achieving a solution to some problem.

Salience of Elements. Models of analogical problem solving distinguish salience using causal structures, which determine the elements important to the specific purpose of an analogy. These are the same causal structures that operationalize their knowledge preserving criteria.

Some of the reviewed models, such as MEDIATOR, NLANG and ARIES, introduce further distinctions.

MEDIATOR organizes its *generalized problem solving episodes* into a generalization hierarchy and uses this hierarchy to direct the search for potentially useful analogs, given the description of a source, in a top-down strategy[KSS85]. Elements in more general episodes, affect more the retrieval and consequently the elaboration, than elements comprised in more specific episodes. Furthermore, elements are distinguished by their types. Matching between goals is attempted first, and only if successful, it allows the matching between other descriptive elements of analog episodes such as objects and involved agents.

NLANG uses a similar preference heuristic. The elements of analogs, which can lead to useful conjectures, are aggregated into structures called *abstractions*[Grein88a,Grein88b]. The employed heuristic favours the selection of a more general over a more specific abstraction, for guiding the mapping between two analogs.

ARIES utilizes the so called *reinforcement cases*. Reinforcement cases are cases where the transformation of plans solving analogous source problems failed to yield an acceptable candidate plan resolving the target problem. Such cases are used to tune the similarity metric of the model so as to become more responsive to elements that have not led to them[Car83].

Aptness of Analogy. Only few of the models of analogical problem solving incorporate aptness measures.

In particular, MEDIATOR, PRODIGY and ARIES measure aptness by relative or absolute numbers of shared goals, states and conditions of two analogs. Their metrics enable the selection of the best among a set of analog sources but have no role in the evaluation of detected analogies, which are assessed solely from the achievement of the goals related to the target problem.

Other models, such as NLANG, base even the selection of sources on heuristics.

2.3.4 Models of Case-Based Reasoning

Models of case-based reasoning are distinguished from other models of analogical problem solving due to their focus on solving problems in the same application domain[Bur88]. Problem solving is interpreted in a broad sense and may include tasks such as goal achievement and classification.

Thus, our review in this area includes models supporting problem solving in its classical sense, such as PARADYME/JULIA[Kol88], SURVER III[WWK88], HYPO[Ash90], CADET[SN91] and CABOT[CFR91] as well as models performing analogical classifications such as Protos[PBH90], FGP[FG91], CBL4[Aha91] and CBR+EBL[CPS91].

Representation. Models of case-based reasoning usually work under the *uniform representation assumption*[Bur88]. In other words, they assume that analogs(i.e. cases) can be described using predefined and standard sets of features.

All the models in our survey rely on frame-based representations. Also, like the models of analogical problem solving, they use special representation primitives to express relations and properties of cases, which are important to analogies. Examples of such special primitives include the *influence relations* in CADET[SN91], the *causality determining rules* in CBR+EBL[CPS91] and the various *equivalence* and *implication* relations in the explanation language of Protos[PBH90].

Mapping Criteria. Due to their representation frameworks, models of case-base reasoning adopt fewer and simpler mapping criteria, in comparison with other models of analogical reasoning.

In particular, most of them restrict mappings to isomorphisms between identical slots of frames describing cases. Structure preserving criteria are useless since cases are always described through standard sets of features.

Furthermore, only two of the reviewed models(Protos and SURVER III) incorporate knowledge preserving criteria. Protos and SURVER III extend mapping so as to consider features, connected by relations, expressing specific types of resemblances between them. Protos introduces 15 such relations which express different forms of interfeature equivalences(e.g. *implies*, *co-occurs*, *is-equivalent-to*[PBH90]). It also provides heuristic

rules for composing these relations in order to detect equivalences. SURVER III uses *synonymy* relations[WWK88].

Saliency of Elements. Three different types of information are utilized by models of case-based reasoning in

estimating the saliency of individual features of cases:

- (1) the involvement of features in structures indicating their importance for the tasks to be resolved;
- (2) a-priori asserted estimates of saliency; and,
- (3) assessments of the results of reasoning sessions, supplied by some external tutor.

Most of the reviewed models use the first type of information and adopt multi-valued, as opposed to two-value, measures, so as to provide fine-grain saliency distinctions.

In a noticeably different way, FGP model realizes saliency through statistical measures. This model estimates the saliency of a feature as the inverse of the entropy of its values, with respect to some other goal-feature, which is considered to be the basis of saliency estimation[FG91]. Roughly speaking, the less diverse the values of a feature, with respect to some goal feature, the more salient this feature is.

Aptness of Analogy. Measures of aptness support only the selection of the best among a set of candidate source cases, in case-based reasoning. Aptness has no role in the selection of alternative mappings between cases, since the adoption of fixed representations usually does not allow more than one such mappings.

Most of the reviewed models adopt numerical similarity measures which linearly aggregate matching scores of individual features.

Two of the reviewed models(i.e. PARADYME/JULIA[Kol88] and HYPO[Ash90]), do not employ aptness measures. Instead, they select the best source case, using heuristics. CADET uses neither aptness measures nor heuristics. Instead of selecting a single best source case, it attempts to combine analogies from all the candidate source cases, while resolving a problem[SN91].

2.4 Elaboration Based on Causal Knowledge(The Determination Approach)

The solutions provided by the reviewed models, regarding the four basic aspects of elaboration are summarized in tables 2.1, 2.2, 2.3 and 2.4.

Model	Predicate Calc.	Frames	Semantic Nets	Prod. Rules	Special Causal Relations
SME	*				
ACME-ARCS	*	*			*
CST	*				
CARL		*	*		*
MACKBETH			*		*
Purpose Directed Analogy					*
MEDIATOR		*			
PRODIGY	*				*
ARCHES			*	*	
NLANG	*				*
ARIES		*			*
Derivational Analogy		*			*
ANA				*	*
ANAGRAM		*		*	
Explicit Planning	*				
CADET		*			*
PARADYME-JULIA		*		*	
CABOT		*		*	
HYPO		*		*	
SURVER III			*		*
CBR+EBL		*		*	
Protos		*		*	
CBL4		*		*	
FGP		*		*	

The basic difference between general and purpose-driven models(i.e. models of analogical problem solving and understanding) is the utilization of causal knowledge determining the elements of analogs, which are important for analogies serving a particular purpose.

The assumption about the existence of such knowledge leads purpose-driven models to specific computational solutions regarding the representation, the mapping and the

salience aspects of elaboration.

Causal knowledge is expressed using particular relations in most of the reviewed models of analogical problem solving (see last column of table 2.1). In essence, these models adopt a uniform representation assumption in expressing this knowledge. This assumption is extended by models of case-based reasoning. Such models assume the entire representation of cases to be uniform.

Model	Cardinality		Type Compatibility				Structure Preservation	
	1:1	1:N	Ident.	Class Ident.	Arg.Arity	Arg. Compatibility	Top Down	Bottom Up
SME	*		*		*	*	*	
ACME-ARCS		*			*	*		
CST		*	*		*	*		
CARL		*		*	*	*	*	
MACKBETH	*				*	*		
Purpose Directed Analogy	*		*	*				
MEDIATOR	*		*					
PRODIGY	*		*					
ARCHES		*	*					
NLANG	*		*					
ARIES	*		*					
Derivational Analogy	*		*					
ANA	*							
ANAGRAM	*		*		*	*	*	
Explicit Planning	*		*					*
CADET	*		*	*		*		
PARADYME-JULIA	*		*					
CABOT	*		*					
HYP0	*			*				
SURVER III	*		*					
CBR+EBL	*		*					
Protos	*							
CBL4	*		*					
FGP	*		*					

Uniform representation enables purpose-driven models to adopt *identity* as the basic criterion of mapping (i.e. elements in descriptions of analogs cannot be mapped unless they are identical). Identity is the most strict among type-compatibility criteria of

Model	Knowledge Preservation			Consistency
	Semantic Similarity	Causal Knowledge	Tutor	Coherency
SME				
ACME-ARCS	*			
CST				*
CARL			*	
MACKBETH			*	
Purpose Directed Analogy		*		
MEDIATOR		*		
PRODIGY		*		
ARCHES		*		
NLANG		*		
ARIES		*		
Derivational Analogy		*		
ANA				
ANAGRAM				
Explicit Planning		*		
CADET				
PARADYME-JULIA				
CABOT				
HYPO				
SURVER III	*			
CBR+EBL				
Protos	*			
CBL4				
FGP				

mapping(i.e. identity of the class of the elements to be mapped, same arity of relations, compatibility of corresponding arguments). Thus, purpose-driven models are able to elaborate analogies faster than general models, which adopt less strict mapping criteria, in order to allow more possible mappings between analogs. This flexibility is necessary for detecting analogies, in cases where the representation of analogs cannot be a-priori standard as when analogs belong to different domains. Greiner in[Grein88a] presents experiments indicating speed increase along with the adoption of more strict criteria of mapping.

Causal knowledge also enables straightforward distinctions between elements of different salience in analogs' descriptions, since it is expressed by specific primitives, which

can be easily identified syntactically. Thus, it becomes the main source of information for the estimation of salience regarding models of analogical problem solving(see the fourth column in table 2.3). Furthermore, it is used for selecting a mapping between analogs. Mappings leading to conjectures achieving the goals of reasoning sessions, are those eventually selected. Thus, measures of aptness may only be needed for selecting among candidate source analogs, considering these models.

Model	Source of Saliense Information				Measure Type	
	Syntax	User Est	Importance Structures	Assessments	two-valued	multi-valued
SME	*					*
ACME-ARCS		*	*			*
CST						
CARL		*	*		*	
MACKBETH		*	*		*	
Purpose Directed Analogy			*		*	
MEDIATOR			*		*	
PRODIGY			*		*	
ARCHES			*		*	
NLANG			*		*	
ARIES				*	*	
Derivational Analogy				*	*	
ANA				*	*	
ANAGRAM						
Explicit Planning			*			
CADET			*		*	
PARADYME-JULIA			*		*	
CABOT				*		*
HYPO			*			
SURVER III						*
CBR+EBL			*	*		*
Protos			*	*		*
CBL4						*
FGP			*			*

Models of analogical understanding are also based on causal knowledge. However, they hypothesize that this knowledge is provided by some tutor cooperating with the system during or before the elaboration of analogies.

The lack of a specific purpose and consequently causal knowledge related to it, makes general models to preclude knowledge preserving criteria and rely on structure preserving and/or consistency criteria for mapping. Furthermore, such models distinguish between elements of analogs of different salience based on syntactic criteria. For instance, SME derives the salience of predicates from the order of their arguments.

The interference between the existence of a specific purpose for analogical reasoning and the employed mapping criteria is explicitly realized by ACME and ARCS. These models use special semantic relations(e.g. *meronymic* and *synonym* relations [THING90]), in reasoning cases exposing a clear pragmatic aim and rely on structure preserving criteria for general forms of reasoning, lacking a precise pragmatic aim[Thag88].

Model	Heuristics-Based	Measure-Based
SME		*
ACME-ARCS		*
CST		*
CARL	*	
MACKBETH		*
Purpose Directed Analogy		
MEDIATOR		
PRODIGY		*
ARCHES		*
NLANG	*	
ARIES	*	
Derivational Analogy		*
ANA		
ANAGRAM		*
Explicit Planning		*
CADET		
PARADYME-JULIA	*	
CABOT		*
HYPO	*	
SURVER III		*
CBR+EBL		*
Protos		*
CBL4		*
FGP		*

Furthermore, the absence of definite criteria about the attainment of goals makes more important the role of aptness measures in general models. Aptness measures are used by these models not only to select among different candidate sources but also among candidate mappings between the elements of two analogs.

2.5 Critique of Purpose-Driven Models

This thesis does not neglect the role of causal knowledge in speeding up elaboration and providing definite criteria for the evaluation stage of analogical reasoning in cases where it is carried out for a well-defined purpose.

However, the very assumption about the availability and certain other assumptions about the nature and the representation of such knowledge, adopted by purpose-driven models (i.e. models of analogical problem solving and understanding) are not always valid. Consequently, the ability of such models to support interdomain analogical reasoning for real applications lacking a well-defined purpose is questionable.

As Stuart Russell points out in [Rus88], the existence of causal knowledge does not necessarily imply its availability. Non availability may be the result of either ignorance about existing causal knowledge or the high cost of pursuing extensive domain analysis for identifying and acquiring it. Nevertheless, in situations lacking such knowledge, analogical reasoning, based on other general criteria, is still valuable.

Another issue is the extent to which causal knowledge remains the same across domains and thus can be used to accomplish interdomain analogical reasoning.

To our knowledge, research efforts aiming at the identification of causal knowledge focus only on single application domains, such as requirements engineering [SM91, MS92, Maid92] or jurisprudence [Ash90]. As these domain-specific research efforts verify, both the type and the granularity of causal knowledge are open research issues [Maid92]. Arguments like this, cannot be convincingly opposed by examples of interdomain causal knowledge, used in demonstrations of models of analogical problem solving [Grein88a]. In fact, even the authors of such models admit that the substance of this knowledge is not fully understood yet [Grein88a, Grein88b].

To the extent that causal knowledge has a not well understood substance and granularity it is uncertain whether the primitives proposed for representing it are appropriate,

especially across different domains. In our view, such primitives do not always have clear semantics, meaningful across domains. Consider that, Porter, Bareiss and Holter suggest that causal implications can be realized by six different relations(i.e. the relations *definitionally entails*, *requires*, *causes*, *enables*, *implies* and *suggests*) expressing different types of definite and empirical implications in weak domains[PBH90]. This representation polymorphy is also evident from the research of Maiden and Sutcliffe[Maid92]. The latter authors are concerned with the identification and representation of causal knowledge for analogies in requirements engineering. They propose yet another set of relation types for expressing important aspects for analogies(not involving any kinds of implications) in the specific domain of their concern(see section 6.3.3).

This uncertainty about the existence, availability and representation of causal knowledge limit the applicability of models of analogical problem solving in complex interdomain applications. This is because such models tune their computational solutions according to specific assumptions(e.g. uniform representation), which can be devalidated across domains.

Also, computational solutions, which hypothesize the presence of an external tutor providing the necessary information for elaborating analogies(e.g. models of analogical understanding), are impractical in cases where such cooperating agents are themselves ignorant of determinants of analogies.

Thus, the approach of general models of analogical reasoning, which instead of assuming specific forms of causal knowledge use domain independent criteria for identifying analogies, is the only one that can offer generally applicable solutions to elaboration. However, certain computational solutions of general models are also weak in dealing with real applications.

2.6 Critique of General Models

Proposed general models of analogical reasoning are weak in three aspects:

- their employed preference criteria;
- their ability to distinguish causal knowledge, whenever this knowledge, albeit embedded in descriptions of analogs, is not explicitly acknowledged as such; and,

- their employed measures of aptness.

In the following, we discuss these weak aspects.

2.6.1 Weakness of Preference Criteria

Evidently from table 2.2, the preference criteria used by general models of analogical reasoning include:

1. the top down structure preserving criterion(SME,ARCS,ACME);
2. the coherency criterion(AST); and,
3. the knowledge preserving criteria(ARCS,ACME).

The merits of the knowledge preserving criteria have been discussed in section 2.4. As pointed out, such criteria are not always operational in different reasoning contexts.

However, even the other preference criteria cannot always provide appropriate solutions to elaboration.

The Structure Preserving Criterion. In spite of the empirical evidence about the plausibility of the structure preserving criterion, regarding the way humans elaborate and understand analogies[GL85,GT86,Gen88a], we believe that this criterion fails to reveal certain kinds of analogies.

As discussed in section 2.3.1, structure preservation requires that if two relations are mapped onto each other, their corresponding arguments must also map. This is an overly strict condition about the configurations of analogs.

We have identified at least one general class of analogs, where it fails to detect existing analogies. This class includes engineering artifacts. In such artifacts, it is possible to identify components serving analogous functional roles, although they are not configured according to the structure preservation criterion.

By way of example, consider the data flow diagrams of figure 2.2. These diagrams present the configuration and communication of the modules of two search programs.

Assume that the first of these programs searches a book in a library. According to it, initially it is necessary to identify some general category of books, that may include the required one and then locate this book among all those belonging to the retrieved

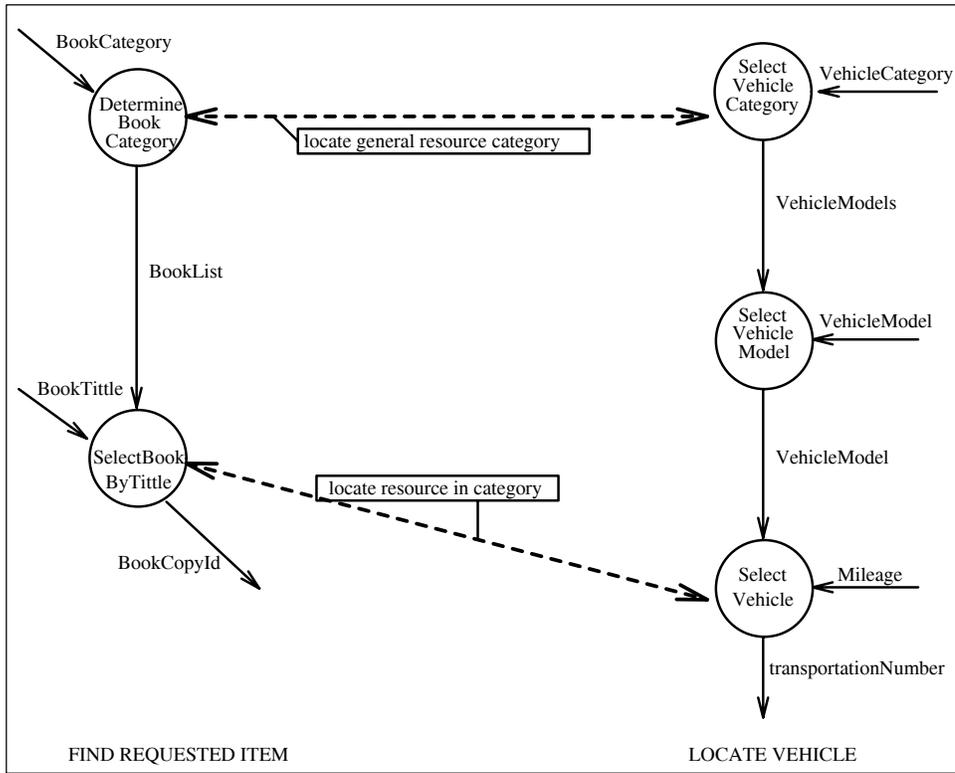


Figure 2.2: Analogies in Searching Programs

category.

Also assume that the second program searches for a car to rent. This program implements a gradual search strategy, too. First, the user has to choose some general type of cars(e.g. cheap cars, jeeps, mini-busses etc.). Then, he must choose a car model in that type. Finally, he has to provide car-specific information(i.e. the mileage) for selecting a specific car of the chosen model and type.

The essence of the analogy between these programs is the gradual search they implement. Initially, they locate a general category of items and eventually, they identify the required item in this category. In addition, the second program involves an intermediate stage, which further restricts the initial category of items(i.e. the selection of a particular car model in some type of cars). Analogous stages are dictated by the dashed lines in figure 2.2.

Elaboration based on the structure preserving criterion would have failed to detect these analogies. It would have mapped the first and the second stages of the first program onto either the first and the second, or the second and the third stages of the second program, respectively. Thus, elaboration restricted by this criterion wouldn't allow the mappings depicted in figure 2.2 and therefore it wouldn't detect the analogy of the gradual search.

The Coherency Criterion. Consistency preserving criteria, like the coherency criterion in CST, are also weak mapping criteria. This is because the consistency of any set of first-order formulas is not decidable. Thus, the criterion is not even computable.

As already mentioned, coherency of AST is an approximation of consistency within some limited resource bounds. However, such approximations neglect the very essence the criterion, since they cannot ensure the consistency of the involved formulas.

2.6.2 Weak Distinctions of Causal Knowledge

General purpose models also fail to cope well with causal knowledge, when this knowledge is present in descriptions of analogs, albeit not explicitly acknowledged as such. This limitation results from the weak salience distinctions between characteristics of analogs.

AST does not address the problem of estimating salience. Also, ACME and ARCS have the same problems with models of analogical problem solving, since they resort on distinctions based on causal knowledge, which is expressed by specific relations asserted by an external tutor(salience is estimated using predefined measures associated with such relations).

Finally, SME detects salience from the *order* of relations in the descriptions of analogs. In particular, relations, functions and attributes of individuals are considered as first-order elements. Relations having as arguments first- order elements are considered as second-order and so on. Because of its top down matching, which is enforced by the structure preserving criterion, SME guides the whole mapping by higher-order elements. Higher-order elements are always mapped before lower-order ones and thus, they gain a higher importance in elaboration than the latter. Lower-order elements are not mapped unless related by higher-order relations. At an extreme, attributes(i.e. lower-order and single argument relations) do not map. We believe that, it is very unusual to identify many second or higher-order relations, meaningful not only across domains but even in a

single domain. Notice also that the order of relations is only a rough criterion for saliency distinctions, sensitive to different equivalent representation choices. These two aspects are exemplified in reference to Gentner’s representation of Rutherford’s analogy between the atom and the solar systems[Gen83,Gen88a].

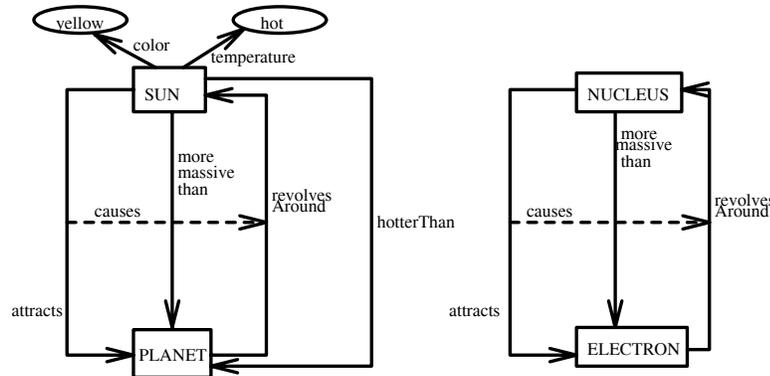


Figure 2.3: The Solar and the Atom Systems of Rutherford’s Analogy

The analogy between the solar and the atom systems, which are presented in figure 2.3, involves the pairwise mapping of the relations *attracts* and *revolvesAround* of the solar and the atom systems. This mapping is derived from the existence of the second-order relation *causes*, between the mapped relations. Unlike the *attracts* and *revolvesAround* relations, other characteristics of the solar system, such as the temperature and the color of the sun are not mapped since they are represented as one-argument relations (i.e. attributes), not involved in any higher-order ones.

However, the temperature could have been modeled as a first-order relation, involved in a second-order one, as shown in figure 2.4. This alternative model presents the higher temperature of the sun, in contrast to the planets, by a second-order relation (i.e. the relation *greaterThan*) over their first-order attributes *temperature*.

Because of their involvement in a second-order relation, the attributes *temperature* would be considered as important as the relations *attracts* and *revolvesAround* and thus, as candidates for mapping[FFG90]. Failing to map them onto identical relations of the atom

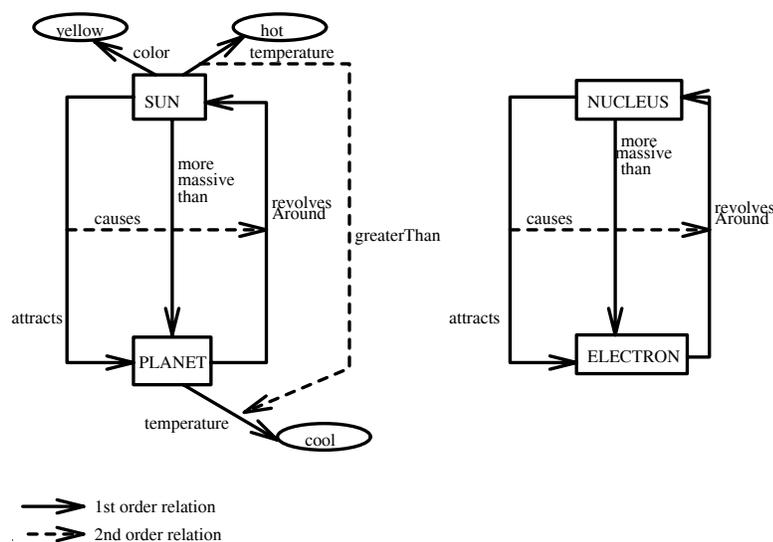


Figure 2.4: An Alternative Modeling of Rutherford's Analogy

system would decrease the aptness of the detected analogies.

Notice also that analogs may have relations of equal order, having different importance to analogies. For example, helms and car-decks, as parts of ferry-boats may both be represented as first-order relations, in the representation framework of SME. However, they do not have the same importance in detecting analogies between ferry-boats and other objects. Helms as parts of their steering systems are important for their function of moving and thus they should be realized as salient elements, when elaborating analogies involving ferry-boats. Unlike them, car-decks are not so important.

2.6.3 Weakness of Aptness Measures

Our last criticism about general models of analogical reasoning regards their employed measures of aptness.

These measures, except in the case of AST, require the provision of explicit matching scores. Matching scores are assigned to mappings according to the specific type of the mapped predicates (e.g. mapping of identical relations in [FFG90] or mapping between semantically similar predicates in [THNG90]). Thus, they cannot really discriminate between mappings of identical or semantically similar relations involving different

arguments. Furthermore, they can only have ad-hoc interpretations (i.e. interpretations not based on well-defined properties, such as the properties of distance or similarity measures [Tve77]). In fact, matching scores usually are not interpreted at all, to our knowledge.

2.7 The Conceptual Modeling Approach to Analogical Reasoning

2.7.1 The Need for an Expressive Conceptual Modeling Framework

Recall that the main weakness of models of analogical problem solving (and case-based reasoning) lies in their assumptions about the availability and the representation of causal knowledge for an effective elaboration of analogies. Possible ignorance, diversity of types and the yet not well understood granularity of causal knowledge devalidate all these assumptions.

This thesis adopts an alternative assumption to overcome the problem. Analogies can be detected through an analysis of conceptual models. The term *conceptual model* is used in the sense of [BMW86]. It denotes human-oriented models of artifacts, consisting of symbolic structures that represent various properties and relations of these artifacts. Symbolic structures are constructed according to semantic modeling abstractions with well-defined semantics. The interpretation of symbolic structures in the descriptions of analogs, according to the semantics of their underlying abstractions, reveals the semantic resemblances between artifacts and thus leads to the detection of analogies between them.

Causal knowledge is assumed to be embedded in descriptions of analogs, although it may not be expressed as such through specific relations. Hence, it must be represented through the same semantic modeling abstractions used for describing any kind of relations in the descriptions of analogs. This assumption implies the use of abstractions with rich semantic content and general enough to express the multitude of relations (causal and non causal), entities and concepts necessary for describing analogs in different application domains.

The validity of the conceptual modeling approach to analogical reasoning depends on two factors. The first concerns the inherent capability of the selected modeling abstractions to express analogies between artifacts. The second concerns the ability of these abstractions to express properties and relations in different domains. The first factor

affects the elaboration of analogies itself, while the second is a prerequisite for elaboration across domains.

Unfortunately, none of these factors can be formally ensured, regardless of the particular semantic modeling abstractions that may be chosen. However, it is possible to assess the merits of specific abstractions with respect to three criteria.

Their ability to express analogies between artifacts can be assessed from their semantics and the extent to which they generalize or directly express, specific relations that have been used for the same purpose in proposed models of analogical reasoning.

Their ability to describe artifacts in different domains can be assessed from their presence in data models and knowledge representation languages. Semantic modeling abstractions, which are common in such models and languages, can be reasonably expected to have a high pragmatic utility as representation devices. Furthermore, their representational adequacy can be evaluated by specific modeling experiences in different application domains, reported in the literature.

In the next section, we introduce three semantic modeling abstractions which, in our opinion, constitute an appropriate representation framework for detecting analogies and building conceptual models of artifacts in different domains. These abstractions are axiomatically defined in the third chapter of this thesis.

2.7.2 Semantic Modeling Abstractions for Elaboration of Analogies

We claim that analogies between artifacts can be detected from their conceptual models built according to the semantic modeling abstractions of classification, generalization and attribution.

Classification. Classification is used for grouping entities and relations, that share common characteristics into classes which aggregate these characteristics [BMW86, PM88]. For instance, descriptions of different persons can be classified under the common class *Person*. Also relations of persons with their fathers and spouses can be classified under the classes of relations *fatherOf* and *spouseOf*. Like individuals, which are grouped into classes, classes are grouped into metaclasses, metaclasses into metametaclasses and so on. Entities and relations can be classified into more than one classes. Multiple classifications are independent from each other, when they express groupings according

to independent characteristics. For instance, persons and companies, in addition to their classification under the classes *Person* and *Company*, can be classified under the class *TaxableEntity*, subject to their obligation to pay taxes.

Usually classification is realized through the relations *kindOf* and *instanceOf* in different semantic and object-oriented data models and knowledge representation languages[Bro84,BMW86,PM88,KL89,My190].

Classification relations are used in almost all the reviewed models of analogical reasoning, though they are not always referred to as such. For example the distinction of elements into goals and conditions in problem-oriented representations of analogs is a classification distinction.

Such relations are utilized by mapping criteria. Two of the type compatibility criteria, *class identity* and *arguments compatibility*, allow mappings between description elements, classified under the same class or having corresponding arguments(if they are n-ary relations), which are classified under the same class, respectively. Notice that classification is widely used because it inherently expresses the most deep type of analogy between objects, the analogy of their substance. As pointed out in[Tur88], common classifications in mental models of humans, express consolidated(i.e. undoubtedly realized) analogies.

Generalization. Generalization is used for extracting from two or more classes their commonalities and aggregating these commonalities into another class [BMW86]. A by-product of generalization is the suppression of the detailed differences of classes. For instance, *Persons* and *Animals* can be generalized into the class *NaturalKindObject*, while *EngineeringArtifacts* and *CulturalArtifacts* can be generalized into the class *NominalKindObject*. Classes of relations can be also generalized. For example, the classes of relations *fatherOf* and *motherOf* between persons can be generalized into the general class of relations *parentOf*.

Generalization is realized through the *Isa* relation in most of the semantic and object-oriented data models[PM88,BK87,LVR88,KL89] and knowledge representation languages[ML84,FK85,My190]. *Isa* relations have been attached different semantics in these models[Bra83]. The most clear semantics is *set inclusion*[BMW86,KMSB89,MBJK90]. According to this, *isa* relations between classes

express that the less general class(i.e. the *subclass*) groups a set of instances, which is a subset of the set of instances grouped by the more general class (i.e. the *superclass*). Thus, the common characteristics of the instances in the superset are also characteristics of the instances in the subset. In this sense, they can be extracted and aggregated into the superclass and consequently be inherited by the subclass. By virtue of the isa relations, subclasses need to retain only the additional common characteristics of their own instances.

Generalization relations are also assumed to indicate analogies in models of analogical reasoning. For instance, ACME, ARCS, MACKBETH, CARL, MEDIATOR and CADET directly use such relations in elaborating mappings. However, generalization expresses a different kind of analogy from classification. It does not reveal an analogy over the substance of objects but indicates semantic resemblances regarding the characteristics of classes having the same substance. In this sense, it clarifies an analogy over the substance(see the discussion in section 2.8.1). Notice, however, that the common substance of two classes related by an isa relation, may not be expressed in a conceptual model through a classification under a common metaclass, either due to incompleteness or due to failure to recognize it.

Attribution. Attribution allows the expression and association of relations and properties(both referred to as *attributes* in the following), with individuals, as well as the expression and association of classes of relations and properties, with classes grouping these individuals. In essence, it enables the description of analogs by enumerating their characteristics in the first place.

As discussed in[MBJK90], attributes may also be associated with other attributes(either individual or classes) so as to express aspects that characterize entire relations rather than any of their involved arguments, separately. Attributes can have their own attributes and be related through classification and generalization relations, exactly like entities[MBJK90].

This uniform view of modeling entities and relations, results into a powerful conceptual modeling mechanism, since it allows the definition of many different kinds of relations. Relations can be defined using only the three mentioned modeling abstractions, without having to supply specific representation primitives for each of them.

Attribution primitives are offered by many semantic data models and knowledge representation languages, as well as by representation frameworks of most of the models of analogical reasoning. Since attributes can have their own attributes, classes and superclasses attribution, as adopted here, can express most of the primitives provided by all these models.

For instance, the different kinds of relations used by the ARCS system to express semantic similarities(i.e. *synonymy*, *hyponymy* and *meronymy*[THING90]) can be defined as different classes of attributes. Also, the *influences*, introduced to express causal implications between quantitative variables in[SN91], can be expressed as in figure 2.5. In chapter 6, we define the meta schema and some of the abstractions, introduced by Neil Maiden[Maid92] to aggregate causal aspects for the detection of analogies in requirements engineering. Our definitions use exactly the modeling constructs of the prescribed framework.

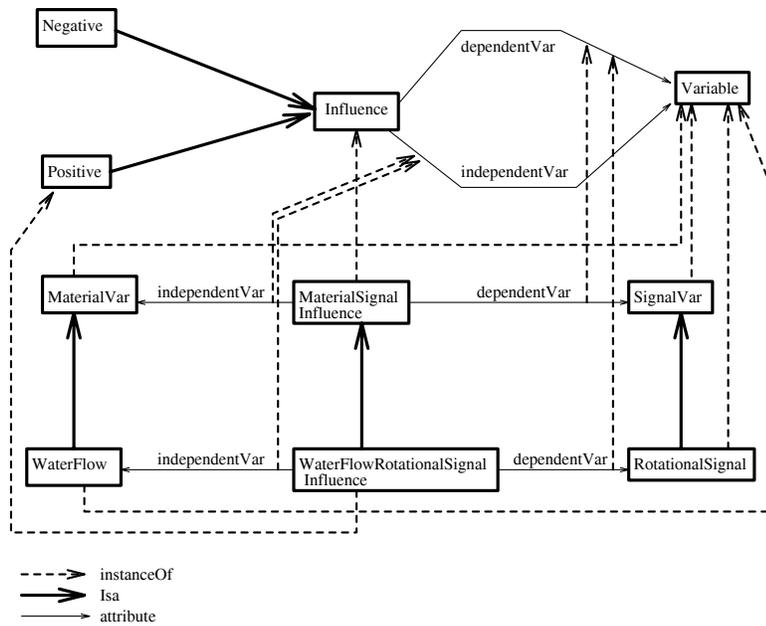


Figure 2.5: A Conceptual Model of Influences

Attribution is extensively utilized by mapping criteria of models of analogical reasoning. Aspects such as the arity or the identity of relations are used as type compatibility criteria.

Notice that classifications and generalizations of attributes reveal analogies between non identical attributes. For instance, it is possible to detect that the non identical classes of attributes *fatherOf* and *spouseOf* have an analogous substance as family relations, by their common classification under the metaclass of attribute classes *familyRelation*.

The three prescribed abstractions constitute a highly expressive framework for describing knowledge related to analogies. The similarity model is defined using this representation framework. This model is presented in the following section and is formally defined in the next two chapters.

2.8 Computation of Analogical Similarities Between Conceptual Models

We introduce a general model of elaborating analogies based on the assumption that analogies can be detected from conceptual models of analogs.

Our model offers certain computational solutions with regard to mapping, salience and aptness, which are novel by comparison to those offered by other general models of analogical reasoning. All these solutions depend on the semantics of the abstractions described in the previous section. We claim that they constitute an elaboration model, which:

- is domain independent;
- is operational even under non uniform representations of analogs;
- realizes human expectations about the detection of analogies; and
- is not intensive to the acquisition of specific relations and/or measures relevant to the detection of analogies or the estimation of salience.

In the following, we discuss the basic aspects of the model and the validity of the previous characterizations.

2.8.1 Distance Metrics and Salience Measuring Functions

Our model comprises three basic metric functions, which measure the distance of conceptual models of analogs with respect to their classification, generalization and

attribution. These functions need no user supplied arguments or predefined distance measures. A fourth metric distinguishes between identical and non identical objects.

These metrics are complemented by another set of functions measuring the salience of attributes of analogs.

The Distance Metrics. Distance metrics were preferred over other forms of matching functions (e.g. the *ratio* or the *contrast* models of similarity [Sjo72,Tve77,Win80,SC93]) since they have clear mathematical properties, namely symmetry and triangularity, with intuitive geometric interpretations.

The partial measures obtained for each of the prescribed semantic modeling abstractions are aggregated into an overall distance measure. This overall measure is transformed into a similarity measure, which reflects the aptness of detected analogies.

Partial measures indicate analogies between conceptual models in different grains.

The distance over classification expresses whether or not the involved artifacts have an analogous substance. The distance over generalization provides a coarse-grain indication of their semantic resemblances. Finally, the distance over attribution clarifies in full detail the analogies between artifacts by establishing a mapping between their analogous attributes.

The role of these metrics is exemplified in figure 2.6.

This figure presents three versions of conceptual models of the solar and the atom systems in Rutherford's analogy[Gen83].

The first conceptual model(see model a in figure 2.6) includes only the classification of the solar and the atom systems under the class *PhysicalSystemClass*. This classification exposes the essential analogy of the involved systems. They both are physical systems composed of physical objects. The computation of their classification distance identifies their common and non common classes and outputs a measure based on the importance of the latter. Assuming that the solar and the atom systems are not classified in any classes in addition to the class *PhysicalSystemClass*, their classification distance will be equal to 0(see definition 3.21 in chapter 3), indicating their identical substance.

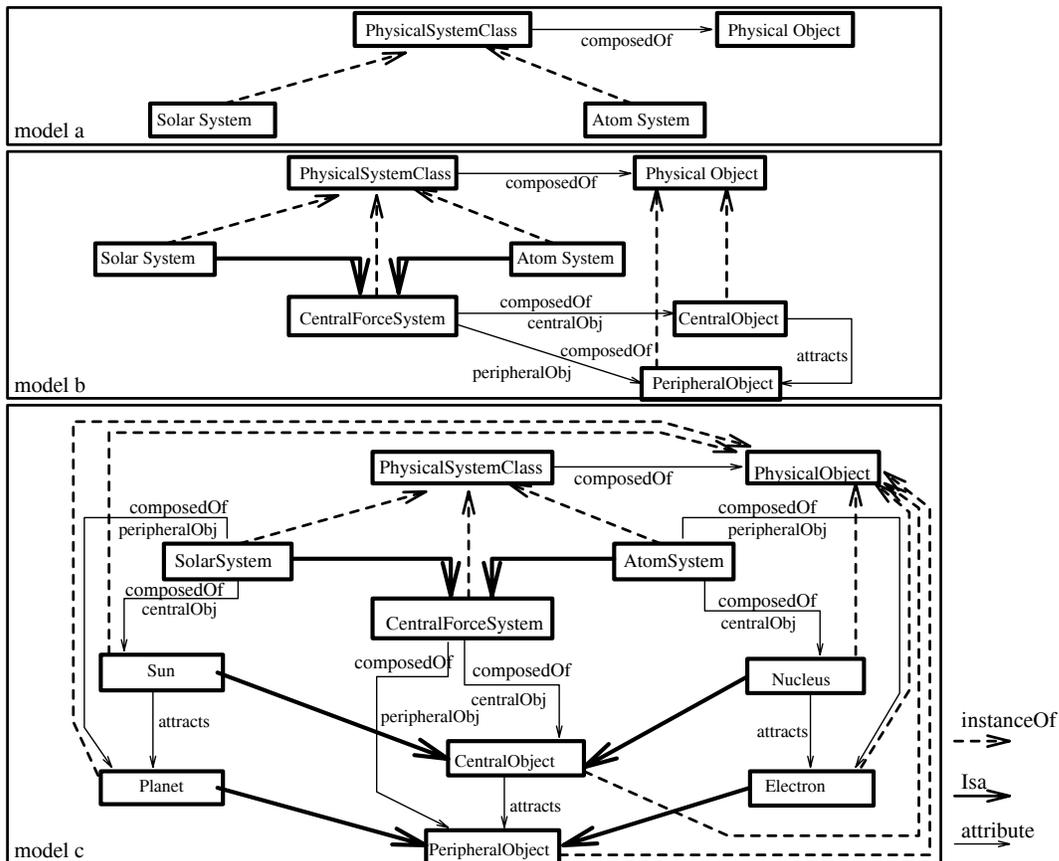


Figure 2.6: Analogies in Increasingly Detailed Conceptual Models

The generalization of the solar and the atom systems to the common superclass *CentralForceSystem*, introduced by the second conceptual model (see model b in figure 2.6), augments the information about their analogy. This model reveals that the involved physical systems are central force systems, which have one or more central objects attracting one or more peripheral ones. The computation of the distance over generalization identifies the common and non common superclasses of analogs and outputs a measure based on the importance of the latter. In reference to model b of figure 2.6, the generalization distance of the solar and the atom systems, would be very close to 0, but not equal to it. This is because the relevant systems are considered as non-shared superclasses of themselves (see definition 3.28 in chapter 3). Such a low generalization distance indicates the presence of only a few semantic differences.

Finally, the detailed attribution of the solar and the atom systems (see model c in figure 2.6), reveals their analogies in full detail. The sun and the nucleus are the analogous central objects, while the planets and the electrons are the analogous peripheral objects. The sun attracts the planets and the nucleus attracts the electrons. The distance over attribution elaborates a mapping between the attributes *peripheralObj* and the attributes *centralObj* of the *SolarSystem* and the *AtomSystem*, as well as between the attributes *attracts* of the *Sun* and the *Nucleus*. This mapping expresses the mentioned analogies.

The three partial distances, also establish the ability of similarity model to reveal analogies from incomplete conceptual models of analogs. As will become evident after the discussion about the criteria of mapping employed by our model (see sections 2.8.2 and 3.3.4), the detection of the analogy between the sun and the nucleus as well as between the planets and the electrons, would be possible even without the presence of the common superclass *CentralForceSystem*. Consider also that conceptual models are finite. Thus, their objects will eventually have no generalization or attribution relations. In such cases their analogies would be approximated by the estimation of the classification distance.

The computations of the prescribed distance metric functions are restricted by a set of principles, which are discussed in the following section.

Salience Measuring Functions. Conceptual models embody attributes referring to elements both important and unimportant to analogies. As discussed in section 2.4, the presence of important elements and their distinction through modeling primitives, tailored for this purpose, is not always possible.

Our model assumes that important attributes of analogs may not be explicitly distinguished as such and consequently they must be identified through criteria, defined over the semantic modeling abstractions, which are used for expressing them.

This approach is justified by the following observation. If certain properties attributes, known to indicate their importance in elaborating analogies, can be associated with specific patterns of representing these attributes in conceptual models, then their importance can be derived from the existence of these patterns[SC94c]. This observation is verified by empirical studies about the formation of mental models by humans. Such studies indicate that models built by experts in some domain are encoded around the

dominant features in these domains[Gen88b].

The similarity model introduces three properties of attributes, which are important to analogies. These properties are:

- i. *charactericity* ;
- ii. *abstractness* ; and
- iii. *determinance* .

Charactericity reflects the ability of an attribute to distinguish between classes in conceptual schemas and to enable the classification of an analog under one of them. Thus, characteristic attributes are important to analogies because they relate to their substance.

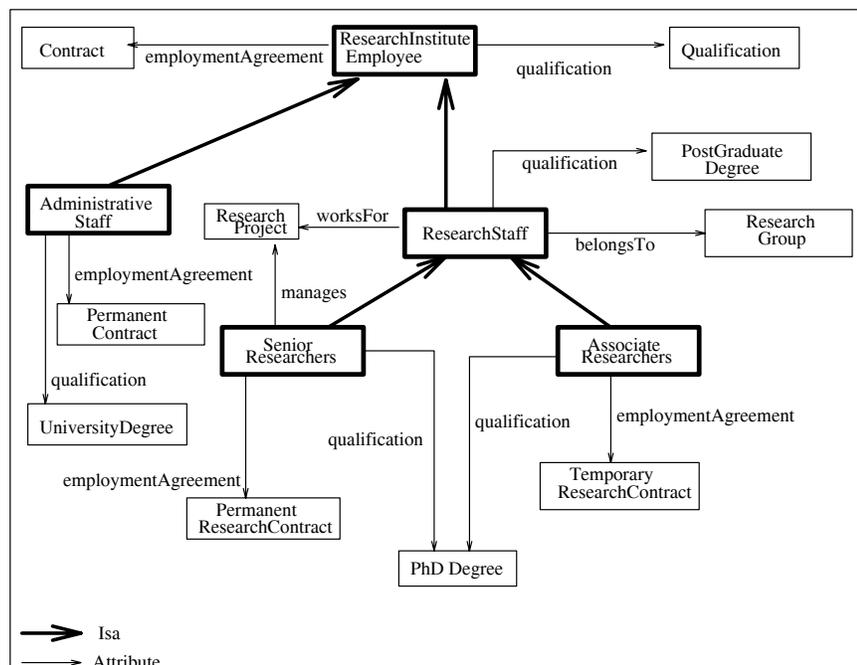


Figure 2.7: Characteristic Attributes

For instance, the attribute *employmentAgreement* distinguishes between the different classes of employees in the taxonomy of figure 2.7 and, determines the classification of each particular employee under a specific class in this taxonomy. In essence, this

attribute determines the legal substance of persons as employees. Thus, it should be given high attention in elaborating analogies between employees and other objects.

Abstractness reflects how essential an attribute may be to the behavior and the existence of the analogs it applies to. For instance, the steering system is essential for any kind of vehicle. Abstract attributes tend to be introduced by the most general classes in conceptual schemas and be inherited by their subclasses. Thus, we would expect the steering system to be introduced by the general class of vehicles and be inherited by all the special kinds of vehicles, such as cars, airplanes, ships and so on.

Determinance reflects the ability of an attribute to determine the values of other attributes. For instance, the qualifications of an employee may determine the position he/she has in an organization. Causal attributes in our approach correspond to characteristics involved in causal implications, which are acquired for elaborating analogies, by most of the purpose-driven models of analogical reasoning(e.g. influences in[SN91], domain rules in[CPS91]). However, they are not recognized due to their involvement in predefined constructs for expressing causal knowledge.

As discussed in chapter 4, the three properties underlying our estimation of salience, are associated with specific patterns of representing attributes in conceptual models. The similarity model introduces functions, that measure the extent to which such patterns occur and uses them to derive belief measures about the truthness of the relevant properties. Such beliefs are interpreted as indicating the salience of attributes.

It must be emphasized that the estimation does not rely on any user supplied or a-priori known salience estimates. However, it can accommodate certain kinds of such estimates, if they are available from the user(see section 4.5 in chapter 4).

2.8.2 Principles of Similarity Computation

The definitions of the partial distance metric functions of similarity model are based on five principles, which constitute a plausible framework for elaborating analogies[SC93]. These principles are:

- (1) the principle of *Ontological Uniformity*;
- (2) the principle of *Isomorphic Analogical Mapping*;

- (3) the principle of *Semantic Homogeneity*;
- (4) the principle of *Minimum Distance Isomorphism*; and,
- (5) the principle of *Normalization*.

The first of these principles restricts the validity of elaboration in analogical reasoning as an operation applicable only to certain kinds of conceptual models. The next three principles introduce the mapping criteria of similarity analysis and the fifth restricts the type of distance measures and their transformation into similarity measures.

The Principle of Ontological Uniformity. According to this principle analogies can exist only between objects having the same basic ontology. In the representation framework of similarity analysis, the most rough distinction about the ontologies of objects is provided by their classification level. Objects may denote individuals or classes of individuals or metaclasses grouping classes of individuals and so on. In this object space, it would be senseless to elaborate analogies between such different objects. For instance, no analogy can exist between the solar system and the class of physical systems in figure 2.6.

Hence, interobject comparisons are restricted according to the principle of the *Ontological Uniformity*:

(P1) Similarity comparisons are only valid between objects having the same classification level.

As will be shown in chapter 3, this principle is enforced by appropriate restrictions over the domains of the distance metric functions.

The principle of ontological uniformity is flexible enough to allow the elaboration of analogies between objects which are not classified under exactly the same classes. This flexibility is necessary for the detection of analogies across domains.

For example, in reference to figure 2.8, it is necessary to allow comparisons between employees and companies so as to find out that they are analogous in the sense that they both have to pay taxes (see the common classification of employees and companies under the class *TaxableEntity*). Notice that, persons and companies also have non common classes (i.e. the classes *Person* and *Company*).

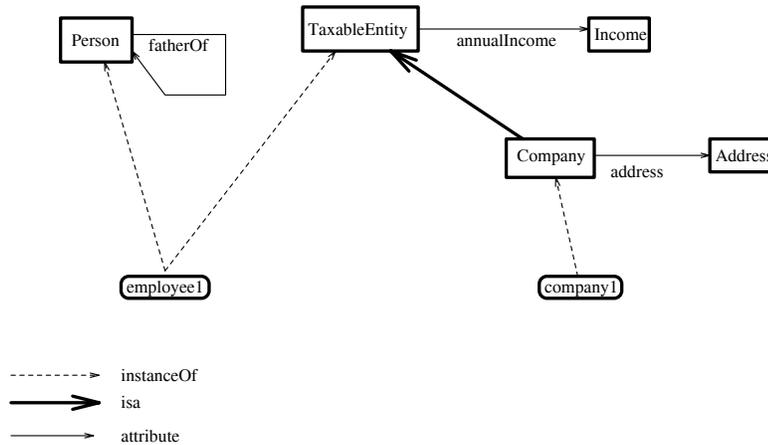


Figure 2.8: *Ontologically Uniform Analogs*

However, it would be senseless to try to elaborate an analogy between the class of persons *Person* and the specific company, *company1*.

The Principle of Isomorphic Analogical Mapping. Cognitive studies have indicated that humans are sensitive to isomorphisms, while elaborating mappings between analogs[GT86,Gen88a, Thag88,THING90].

This empirical finding corresponds to the intuition that analogous elements of analogs must be configured isomorphically, in their conceptual models. Accordingly, our model like most of the reviewed ones(see table 2.2), restricts the possible mappings between analogs by the principle of *Isomorphic Analogical Mappings*:

(P2) Analogical mappings between attributes of objects must be isomorphic.

This principle is realized by the definition of the attribution distance, which considers only 1:1 mappings, when searching for the most plausible mapping between the elements of two analogs.

The Principle of Semantic Homogeneity. The principle of Semantic Homogeneity is introduced due to the fact that analogs from different domains are rarely described through a fixed, predefined set of properties and/or relations. In other words, they may have non uniform representations. Thus, it is necessary to adopt type compatibility

criteria for restricting the mappings between their different description elements. Such criteria replace the identity based mapping, that can be used under the uniform representation assumption, with criteria defining equivalence relations between attributes[Bur88].

The principle of Semantic Homogeneity is an equivalence, type compatibility criterion:

(P3) Attributes cannot be mapped onto each other, unless they belong to the same classes.

According to our interpretation of classification, this principle allows mappings only between attributes having the same substance.

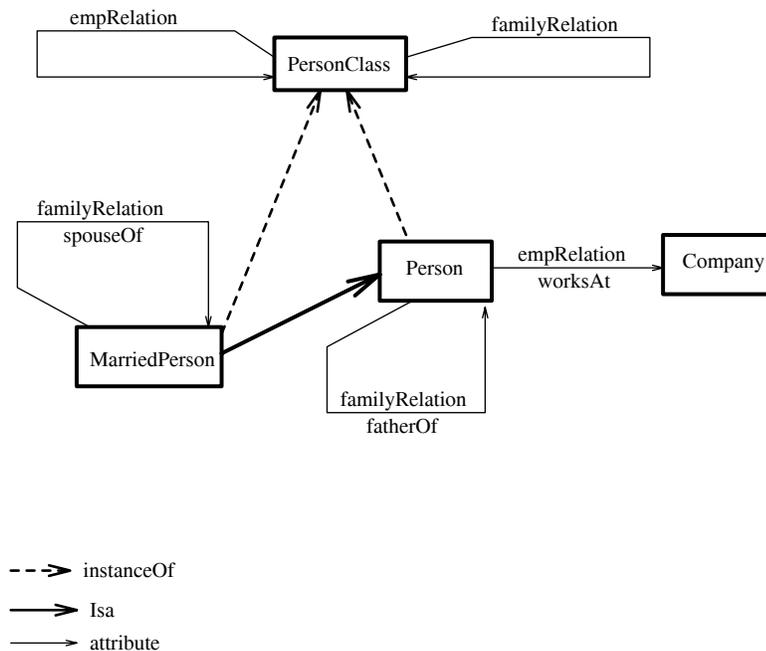


Figure 2.9: Semantically Homogeneous Attributes

For example, the relations *fatherOf* and *spouseOf* of the classes *Person* and *MarriedPerson* in figure 2.9, can be mapped onto each other since they are both classified under the common attribute meta class *familyRelations*. However, none of them could be mapped onto the relation *worksAt*, since it has been classified as an employment rather than a family relation.

The principle of semantic homogeneity is a domain independent criterion since it is not defined on the basis of any domain-specific information.

The Principle of the Minimum Distance Isomorphism. This principle provides a preference criterion for selecting among alternative isomorphisms between the attributes of two analogs:

(P4) An analogy between two objects is expressed by the isomorphism between their attributes, which has the smallest overall distance.

The overall distance of an isomorphism is estimated by aggregating the partial distances of each of its constituent pairs. Notice that although some of the attributes of two objects, may not be mapped onto the less distant attributes to themselves, this principle ensures that the overall isomorphism will be optimal.

Thus it provides an objective criterion, which ensures that no other isomorphisms between the attributes of two analogs can be more coherent than the selected one.

This criterion is more powerful than the structure preserving criteria adopted by other general models of analogical reasoning in selecting among competing isomorphisms between analogs. This is because it is based on the entire semantic similarity of the elements to be mapped rather than on the configuration of these elements in the involved analogs.

For example, assuming the absence of the *attracts* relation between the sun and the planets and between the nucleus and the electrons in model c of figure 2.6, structure preserving criteria, could have mapped the sun to the electron and the nucleus to the planet.

The minimum distance criterion overcomes this problem. It does not only take into account the structural configurations of the relations *centralObj* and *peripheralObj* in the solar and the atom systems. These configurations would be preserved regardless of whether the *centralObj* relation of the *SolarSystem* would have been mapped onto the *peripheralObj* relation of the *AtomSystem*. The principle of the minimum distance isomorphism solves the problem by considering the overall semantic resemblances between these relations. Thus, it can detect that the sun resembles the nucleus (both are special kinds of central objects) more than the electron, which in turn resembles more the planet

than the sun(both are special kinds of peripheral objects). Consequently, it maps the relevant relations onto each other. In a similar manner, it leads to a correct mapping between the stages of the data flow diagrams of figure 2.2, as explained in detail in chapter 6.

Like the principles of isomorphic mapping and semantic homogeneity, the principle of the minimum distance isomorphism is also domain independent, since distances between pairs constituting isomorphisms are not computed in a domain dependent way, as will become evident in the chapter 3.

The Normalization Principle .

(P5) Similarity must be a monotonically decreasing function of a normalized distance measure.

The monotonically decreasing behavior of similarity is evident in experiments about how humans assess similarity with respect to observed distances between objects[Rus88]. Also, normalization enables the filtering of differences which arise due to coarser and finer representations of analogs(e.g. attribute ranges with unequal widths, diverse coarsenesses of Isa-graphs).

2.9 Summary

Analogical reasoning has been computationally supported by various models proposed in the literature. These models are distinguished by the purpose for which such reasoning is performed into general models, models of analogical understanding and models of analogical problem solving. The latter include the models of case-based reasoning, which perform analogical problem solving in single domains.

Models of the last three types base the elaboration of analogies on causal knowledge determining the features of analogs, which are important to them. Such models are weak in supporting interdomain analogical reasoning because, usually, their assumptions about the availability and the representation of causal knowledge are not valid across domains.

Also, general models are weak in identifying such knowledge, whenever present, and adopt preference criteria not always effective for the elaboration of analogies.

We introduce a general model of elaborating analogies, which adopts a conceptual modeling approach and elaborates analogies based on distance and salience measuring

functions, which are defined over three semantic modeling abstractions (classification, generalization and attribution). These abstractions enable the description of analogs from different domains and inherently express analogies between them. The definitions of the functions of the model, which are presented in the next two chapters, satisfy a set of five principles, which guarantee the domain independence of the model and its operation over non uniform descriptions of analogs.

The model employs no user-supplied or a priori asserted distance and salience measures so as to be operational in application domains and tasks, where such measures are not readily available, and to avoid the inevitably ad-hoc interpretations of such measures.

Chapter 3

Distance and Similarity Functions

3.1 Introduction

In this chapter we define and discuss the properties of the distance and similarity functions which operationalize the conceptual modeling approach of the similarity model. An absolute and a relative function are introduced for each of the basic conceptual modeling abstractions, which, as discussed in the previous chapter, provide a device for the analysis of analogical similarity. These functions are complemented by a distance over the identification of objects. Thus, the whole model consists of:

- a) an identification distance (i.e. the function d_1);
- b) an absolute and a relative classification distance (i.e. the functions D_2 and d_2);
- c) an absolute and a relative generalization distance (i.e. the functions D_3 and d_3);
- d) an absolute and a relative attribution distance (i.e. the functions D_4 and d_4);
- e) an aggregate overall distance measure (i.e. the function D); and,
- f) a similarity measure, inverting the overall distance measure(i.e. the function S).

The functions d_1, d_2, d_3 and d_4 are aggregated into an overall distance measure.

Before, the discussion of these functions, we define the conceptual modeling framework for the analysis of similarity.

3.2 The Conceptual Modeling Framework for Similarity Analysis

Overview. The proposed conceptual modeling framework is essentially a structurally object-oriented data model. Objects are structured through three basic abstraction mechanisms, namely attribution, generalization and classification.

Attribution allows the association of an object, with attribute objects, which reflect its various characteristics (i.e. properties, relations etc.). Attributes, being objects themselves, can have their own attributes, and also be structured according to the other two abstractions, in the framework.

Classification is used to group objects that share common characteristics into classes. Classification has a *set membership* semantics. Thus classes are interpreted as sets, which are defined by an enumeration of their elements. This is possible by declaring an object as an *instance* of a class. Classes abstract the common characteristics of their instances into classes of attributes, which in turn group the attribute objects that reflect these characteristics. Thus, they provide generic descriptions for their instances, playing a role analogous to frames and object types in frame-based[BFL83,ML84,FK85] and object oriented systems[MSOP86,AH87,BK87,LRV88,KL89], respectively. Classification is infinite: atomic objects (also called *Tokens*) are classified into classes classes into meta classes, meta classes into meta meta classes and so on.

Generalization allows the extraction of the common characteristics of classes and their association with more general classes, which are declared as superclasses of the former (with the use of the *Isa* relation). Generalization has a *subset* semantics. Thus, superclasses are interpreted as supersets of their subclasses. This semantics justifies the *strict inheritance* of attributes, which are associated with superclasses, to their subclasses. In fact, inheritance occurs since a property(i.e. an attribute), which holds for all the elements of a set (i.e. the superclass) also holds for any of its subsets(i.e. the subclass). It is possible for a class to have more than one superclasses, not connected themselves by an isa relation. Also, attribute classes may be generalized into attribute superclasses.

These abstraction mechanisms provide a really expressive framework for representing knowledge about entities. A distinct advantage of this framework by comparison to other object-oriented [MSOP86,AH87,BK87,LRV88,KL89] and semantic data models [PM88], is the multiple and infinite classification. This realization of classification makes possible the definition of different meta models which consequently can be used for building conceptual models, according to their semantics. This is because it is also possible to abstract semantic properties of the relationships and the entities comprising these conceptual models, into appropriate classes. Another powerful aspect of this framework is the realization of attribution. This realization makes possible not only the representation of very diverse kinds of relationships, but also the abstraction of commonalities between such relationships using the other two abstraction mechanisms. Thus, there exists a large flexibility in expressing various kinds of semantics and in abstracting commonalities between objects, which are prerequisites for the assessment of analogical similarity, as argued in the previous chapter.

Origins. The prescribed conceptual modeling constructs are offered by the structural part of the Telos knowledge representation language [KMSB89,MBJK90], descending from RML [Gre84] and CML [Sta86].

The interested reader may refer to [KMSB89] for a formalization of the language based on logic and to [Ple90] for its ontology along with a formal semantics for its temporal dimension. Evidence about the potential of the prescribed abstractions as constructs appropriate for complex knowledge representation tasks has been acquired through the application of Telos to tasks related to the specification and development of Information Systems discussed in [MBJK90,CCD92,JMSV92,CD93,CDV93].

In the following section, we define in a set theoretic manner, the concepts of this modeling framework and present an axiomatic foundation of valid conceptual models, built according to it. These concepts and axioms will be the basis for the subsequent treatment of the similarity model as well as for the proofs of its properties.

3.2.1 Set Theoretic Foundation of Basic Concepts

Basic Definitions. Each object base is associated with two distinct sets of symbols, called object identifiers and logical names.

Definition 3.1: *I is a countably infinite set of symbols called object identifiers. I contains the special symbol nil, called the dangling identifier*

Definition 3.2: *L is a countably infinite set of symbols called object logical names*

Object identifiers are provided internally and allow unambiguous internal references to objects(see definition 3.10 below, also[KC86]). On the other hand, logical names are associated with objects externally by users to accomplish logical references to them.

Our conceptual modeling framework offers primitive types of values defined as:

Definition 3.3: *Let B_p denote a finite set of built in primitive types. A built in primitive type is a 4-tuple*

$$O_{pt} = [pt, l, In, Isa]$$

where

pt is an object identifier ($pt \in I$)

l is an object logical name ($l \in L$)

In is a finite set of object identifiers denoting the classes of O_{pt} ($In \subseteq I$)

Isa is a finite set of object identifiers denoting the superclasses of O_{pt} ($Isa \subseteq I$)

For each primitive type O_{pt} , let $symbol(pt)$ be a set of symbols, called the valid symbols of O_{pt} .

The valid symbols of each primitive type correspond to primitive values of this type, defined as:

Definition 3.4: *A built in primitive value pv is defined as a tuple*

$$pv = [pt, v]$$

where

pt is an object identifier denoting the built in primitive type of the value

v is a symbol in the set of valid symbols of pt, $v \in symbol(pt)$

The primitive values can be regarded as identifiers of objects, which cannot have attributes. The set of all the available primitive values is defined as:

Definition 3.5: *PV is the set of valid primitive values defined as*

$$PV = \left\{ [pt, v] \mid (\exists x_1 : (x_1 \in B_p) \text{ and } (pt = x_1.id) \text{ and } (v \in symbol(x_1.pt))) \right\}$$

Non-primitive objects are defined as:

Definition 3.6: *An object is a 7-tuple*

$$O_{id} = [id, l, FROM, In, Isa, A, TO]$$

where

id is the object identifier of O_{id} ($id \in I$)

l is an object logical name of O_{id} ($l \in L$)

FROM is an object identifier denoting the possessing object of O_{id} ($FROM \in I$)

In is a finite set of object identifiers denoting the classes of O_{id} ($In \subseteq I$)

Isa is a finite set of object identifiers denoting the superclasses of O_{id} ($Isa \subseteq I$)

A is a finite set of object identifiers denoting the direct attributes of O_{id} ($A \subseteq I$)

TO is an object identifier denoting the value object of O_{id} ($TO \in I \cup PV$)

Objects are partitioned into a set of so called *Basic Classes* which also allow their structuring in the first place. Basic classes are defined as:

Definition 3.7: B_w is a finite set of built in objects, called the *Basic Classes*, defined by enumeration as:

$$B_w = \left\{ O_{\#O}, O_{\#I}, O_{\#A}, O_{\#C}, O_{\#W} \right\}$$

where

$$O_{\#O} = [\#O, Object, nil, \left\{ \#W \right\}, \emptyset, \emptyset, nil]$$

$$O_{\#I} = [\#I, Individual, nil, \left\{ \#W \right\}, \left\{ \#O \right\}, \emptyset, nil]$$

$$O_{\#A} = [\#A, Attribute, \#O, \left\{ \#W \right\}, \left\{ \#O \right\}, \emptyset, \#I]$$

$$O_{\#C} = [\#C, Class, nil, \left\{ \#W \right\}, \left\{ \#O \right\}, \emptyset, nil]$$

$$O_{\#W} = [\#W, ObjectClass, nil, \left\{ \#W \right\}, \emptyset, \emptyset, nil]$$

It also offers, another set of classes which group objects belonging to the same instantiation levels, defined as:

Definition 3.8: B_i is a countably infinite set of built in objects, called *Instantiation Level Classes*, defined by enumeration as:

$$B_i = \left\{ O_{\#C0}, O_{\#C1}, O_{\#C2}, O_{\#C3}, O_{\#C4}, \dots \right\}$$

where

$$O_{\#C0} = [\#C0, \text{Token}, \text{nil}, \left\{ \#C1, \#O, \#C \right\}, \emptyset, \emptyset, \text{nil}]$$

$$O_{\#C1} = [\#C1, \text{S_Class}, \text{nil}, \left\{ \#C2, \#O, \#C \right\}, \emptyset, \emptyset, \text{nil}]$$

$$O_{\#C(k+1)} = [\#C(k+1), M(k)_Class, \text{nil}, \left\{ \#C(k+2), \#O, \#C \right\}, \emptyset, \emptyset, \text{nil}] \quad k=1, 2, 3, \dots$$

On the basis of these definitions, an object base is defined as:

Definition 3.9: An object base T is a countably infinite set of objects defined as

$$T = B_w \cup B_i \cup B_u$$

where B_u is a finite set of objects, called *User Objects*.

Objects in an object base are uniquely identified by their object identifiers, due to the existence of an isomorphism, defined as:

Definition 3.10: o is defined as an onto isomorphism

$$o: I \rightarrow T$$

such that for all x_1, x_2 such that $x_1 \in I$ and $x_2 \in T$:

$$o(x_1) = x_2 \iff x_2.id = x_1$$

The isomorphism o is internally constructed by enumeration, upon the creation of new user objects.

Objects are also associated with logical names, through a mapping, defined as:

Definition 3.11: n is defined as an onto, $M: I$ mapping

$$n: I \rightarrow L$$

The mapping n is constructed upon the external declaration of logical names for user objects, while built in objects have reserved logical names.

Class objects have an extension, including all their instances, defined as:

Definition 3.12: *The extension of a class object, with identifier $\#i$, is a set of object identifiers, defined as:*

$$EXT[\#i] = \left\{ x \mid (\#i \in o(x).In) \right\}$$

Objects have their own attributes, which are said to belong to their *intensions*. Intensions of classes comprise their own attributes and the attributes which these classes inherit from their superclasses. Intensions are defined as follows:

Definition 3.13: *The intension of an object, with identifier $\#i$, $INT[\#i]$, is a set of attribute object identifiers, defined as:*

$$INT[\#i] = \begin{cases} o(\#i).A & \text{if } (\#i \in EXT[\#C0]) \\ o(\#i).A \cup INH[\#i] & \text{otherwise} \end{cases}$$

where

$INH[\#i]$ is called the set of the inherited attributes of $\#i$, defined as:

$$INH[\#i] = \left\{ x \mid \left(\begin{aligned} &(\exists x_1 : (x_1 \in I) \text{ and } (x_1 \in o(\#i).Isa) \text{ and } (x \in o(x_1).A) \text{ and} \\ &(\text{not}(\exists x_2 : (x_2 \in I) \text{ and } (x_2 \in o(\#i).A) \text{ and } (x \in o(x_2).Isa) \text{ and } (n(x) = n(x_2)))) \text{ and} \\ &(\text{not}(\exists x_3, x_4 : (x_3 \in I) \text{ and } (x_4 \in I) \text{ and } (x_3 \in o(\#i).Isa) \text{ and } (x_1 \in o(x_3).Isa) \text{ and} \\ &(x_4 \in o(x_3).A) \text{ and } (x \in o(x_4).Isa) \text{ and } (n(x_4) = n(x)))) \end{aligned} \right\}$$

Attributes have also an original class(i.e. their most general superclass which has an identical logical name with themselves) defined as:

Definition 3.14: *The original class of an attribute $\#i$, $OC_{\#i}$ is a class u satisfying the following condition*

$$(u \in o(\#i).Isa) \text{ and } (n(\#i) = n(u)) \text{ and } (\text{not}(\exists x_1 : (x_1 \in o(\#i).Isa) \text{ and } (x_1 \in o(u).Isa) \text{ and } (n(\#i) = n(x_1))))$$

3.2.2 Axiomatic Foundation of Valid Object Bases

An object base T is valid as long as it does not violate any of the following axioms.

i. Axioms of Classification

Axiom A.3.1: *Any user object must be classified under an Instantiation Class.*

$$\forall x_1 : (x_1 \in B_u) \Rightarrow (\exists x_2, x_3 : (x_2 \in B_i) \text{ and } (x_3 \in I) \text{ and } (x_3 = x_2.id) \text{ and } (x_3 \in x_1.In))$$

Axiom A.3.2: *Any non Basic Class object can be classified under only one Instantiation Class.*

$$\forall x_1, x_2 : (x_1 \in (B_i \cup B_u) \text{ and } (x_2 \in B_i) \text{ and } (x_3 \in I) \text{ and } (x_3 = x_2.id) \text{ and } (x_3 \in x_1.In) \Rightarrow$$

$$(not(\exists x_4, x_5 : (x_4 \in B_i) \text{ and } (x_5 \in I) \text{ and } (x_5 \neq x_3) \text{ and } (x_5 = x_4.id) \text{ and } (x_5 \in x_1.In)))$$

Axiom A.3.3: *A user object cannot be classified under another user object, unless its Instantiation level class is an instance of the Instantiation Level Class of this latter object.*

$$\forall x_1, x_2, x_3, x_4 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (x_3 \in B_i) \text{ and } (x_4 \in B_i) \text{ and} \\ (x_3.id \in x_1.In) \text{ and } (x_4.id \in x_2.In) \text{ and } (x_2.id \in x_1.In) \Rightarrow (x_4.id \in x_3.In)$$

Axiom A.3.4: *Any user object, which is not a Token is a Class.*

$$\forall x_1 : (x_1 \in B_u) \text{ and } (not(\#C \in x_1.In)) \Rightarrow (\#C \in x_1.In)$$

Axiom A.3.5: *Any user object must be an instance of Individual or Attribute.*

$$\forall x_1 : (x_1 \in B_u) \Rightarrow ((\#I \in x_1.In) \text{ and } (not(\#A \in x_1.In))) \text{ or } ((not(\#I \in x_1.In)) \text{ and } (\#A \in x_1.In))$$

Axiom A.3.6: *Any user object which is an instance of Individual cannot be an instance of another object, unless that is also an instance of Individual.*

$$\forall x_1, x_2 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#I \in x_1.In) \text{ and } (x_2.id \in x_1.In) \Rightarrow (\#I \in x_2.In)$$

Axiom A.3.7: *Any user object which is an instance of Attribute cannot be an instance of another object, unless that is also an instance of Attribute.*

$$\forall x_1, x_2 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#A \in x_1.In) \text{ and } (x_2.id \in x_1.In) \Rightarrow (\#A \in x_2.In)$$

ii. Axioms of Generalization/Specialization

Axiom A.3.8: *The Isa relation is not valid for Tokens.*

$$\forall x_1 : (x_1 \in B_u) \text{ and } (\#C 0 \in x_1.In) \Rightarrow (x_1.Isa = \emptyset)$$

Axiom A.3.9: *The Isa relation between classes is transitive.*

$$\forall x_1, x_2, x_3 : (x_1 \in T) \text{ and } (x_2 \in I) \text{ and } (x_3 \in I) \text{ and } (x_2 \in x_1.Isa) \text{ and } (x_3 \in o(x_2).Isa) \Rightarrow (x_3 \in x_1.Isa)$$

Axiom A.3.10: *The Isa relation between classes is acyclic.*

$$\forall x_1, x_2 : (x_1 \in I) \text{ and } (x_2 \in I) \text{ and } (x_2 \in o(x_1).Isa) \Rightarrow (\text{not}(x_2 \in o(x_1).Isa))$$

Axiom A.3.11: *The Isa relation between classes has a subset semantics.*

$$\forall x_1, x_2, x_3 : (x_1 \in T) \text{ and } (x_2 \in I) \text{ and } (x_3 \in I) \text{ and } (x_2 \in x_1.In) \text{ and } (x_3 \in o(x_2).Isa) \Rightarrow (x_3 \in x_1.In)$$

Axiom A.3.12: *Every user class object, has as a superclass the Instantiation Level class, which is an instance of its own Instantiation Level Class.*

$$\forall x_1, x_2, x_3 : (x_1 \in B_u) \text{ and } (x_2 \in B_i) \text{ and } (x_3 \in B_i) \text{ and } (x_2.id \in x_1.In) \text{ and } (x_2.id \in x_3.In) \Rightarrow (x_3.id \in x_1.Isa)$$

Axiom A.3.13: *An attribute Class cannot be a specialization of another attribute Class unless its possessing and value objects are specializations of the possessing and value objects of this latter class, respectively.*

$$\forall x_1, x_2 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#A \in x_1.In) \text{ and } (\#A \in x_2.In) \text{ and} \\ (x_2.id \in x_1.Isa) \Rightarrow (x_2.FROM \in o(x_1.FROM).Isa) \text{ and } (x_2.TO \in o(x_1.TO).Isa)$$

iii. Axioms of Attribution

Axiom A.3.14: *Objects, which are instances of the Individual Basic Class have neither a possessing nor a value object.*

$$\forall x_1 : (x_1 \in T) \text{ and } (\#I \in x_1.In) \Rightarrow (x_1.FROM = nil) \text{ and } (x_1.TO = nil)$$

Axiom A.3.15: *Objects which are instances of the Attribute Basic Class must have a possessing and a value object.*

$$\forall x_1 : (x_1 \in T) \text{ and } (\#A \in x_1.In) \Rightarrow \\ (\exists x_2, x_3 : (x_2 \in I) \text{ and } (x_3 \in I) \text{ and } (x_1.FROM = x_2) \text{ and } (x_1.TO = x_3) \text{ and } (x_2 \neq nil) \text{ and } (x_3 \neq nil))$$

Axiom A.3.16: *User objects, which are instances of the Attribute Basic class cannot be instances of a class, unless their possessing and value objects are themselves instances of the possessing and the value objects of this class.*

$$\forall x_1, x_2: (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#A \in x_1.In) \text{ and } (x_2.id \in x_1.In) \Rightarrow \\ (x_2.FROM \in o(x_1.FROM).In) \text{ and } (x_2.TO \in o(x_1.TO).In)$$

Axiom A.3.17: *An attribute object cannot be possessed by more than one objects.*

$$\forall x_1 : (x_1 \in B_u) \text{ and } (\#A \in x_1.In) \Rightarrow (\text{not}(\exists x_2 : (x_2 \in B_u) \text{ and } (x_1.id \in x_2.A) \text{ and } (x_2.id \neq x_1.FROM)))$$

Axiom A.3.18: *The original class of an attribute object is unique.*

$$\forall x_1, x_2 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#A \in x_1.In) \text{ and } (\#A \in x_2.In) \text{ and } (x_2 = OC_{x_1}) \Rightarrow \\ (\text{not}(\exists x_3 : (x_3 \in B_u) \text{ and } (\#A \in x_3.In) \text{ and } (x_3 = OC_{x_1}) \text{ and } (x_3.id \neq x_2.id)))$$

Axiom A.3.19: *Two attribute objects of the same possessing object cannot have the same original class.*

$$\forall x_1, x_2 : (x_1 \in B_u) \text{ and } (x_2 \in B_u) \text{ and } (\#A \in x_1.In) \text{ and } (\#A \in x_2.In) \text{ and } (o(x_1).FROM = o(x_2).FROM) \Rightarrow \\ OC_{x_1} \neq OC_{x_2}$$

3.3 Partial Distance Metrics

This section begins with the formal definition of the basic partitions of the objects in an object base, which are necessary for the subsequent definitions of the distance functions, according to the principle of the *ontological uniformity*. Recall from the previous chapter that, according to this principle, similarity comparisons are not allowed between objects which have different basic ontologies, as they are expressed by the classification level of objects.

In conceptual modeling a coarse distinction between ontologies is made according to two criteria. The first of them is whether an object represents an entity or a relationship in the real world. The second distinguishes between objects representing atomic concepts and objects representing groups of concepts. The validity of these two criteria as a basis for a primitive ontological distinction is justified by the presence of corresponding modeling constructs, in most of the object oriented and the semantic data models.

We realize these criteria, through the following two enforced partitions of objects:

- i) the partition of objects into individuals and attributes; and,
- ii) the partition of objects into successive instantiation levels.

The first of these partitions corresponds exactly to the first criterion of ontological distinctions and it consists of the sets IU and AU, which are formally defined as:

Definition 3.15 : *The set of Individual user objects, IU, is defined as:*

$$IU = EXT[\#I] \cap \left\{ x_1 \mid (\exists x_2 : (x_2 \in B_u) \text{ and } (x_1 = x_2.id)) \right\}$$

Definition 3.16 : *The set of Attribute user objects, AU, is defined as:*

$$AU = EXT[\#A] \cap \left\{ x_1 \mid (\exists x_2 : (x_2 \in B_u) \text{ and } (x_1 = x_2.id)) \right\}$$

The second partition refines the second criterion of ontological distinctions by distinguishing between atomic concepts, groups of atomic concepts, groups of groups of atomic concepts and so on. This partition consists of the sets C_i , defined as:

Definition 3.17 : *The sets of user token objects, simple class objects, meta class objects, meta meta class objects,...($C_0, C_1, C_2, C_3, \dots$) are defined as:*

$$C_i = EXT[\#C_i], i = 0, 1, 2, \dots$$

The combination of these two criteria, results in another partition consisting of the sets $E C_i$ and $A C_i$, defined as:

Definition 3.18 : *The sets of user individual tokens, individual simple classes, individual meta classes, individual meta meta classes,... ($IUC_0, IUC_1, IUC_2, IUC_3, \dots$) are defined as:*

$$IUC_i = IU \cap C_i, i = 0, 1, 2, \dots$$

Definition 3.19 : *The sets of user attribute tokens, attribute simple classes, attribute meta classes, attribute meta meta classes,... ($AUC_0, AUC_1, AUC_2, AUC_3, \dots$) are defined as:*

$$AUC_i = AU \cap C_i, i = 0, 1, 2, \dots$$

Evidently from the subsequent sections, the cartesian products IUC_i^2 and AUC_i^2 ($i=0, 1, 2, \dots$) are the domains of all the distance and the similarity functions.

In essence similarity is only defined between objects, which have the same instantiation level and furthermore they both represent individuals or attributes(i.e. ontologically uniform objects).

Ontologically uniform objects are compared using four partial distance functions. The first of them, i.e. the identification distance, distinguishes between identical and non identical objects. The second, i.e. the classification distance, gives a rough estimate about the analogy of the substance of two objects by measuring and aggregating the importance of all their non common classes. The third, i.e. the generalization distance, gives a rough estimate about the semantic differences of two objects by measuring and aggregating the importance of all their non common superclasses. The fourth, i.e. the attribution distance, measures the detailed differences regarding the attributes of the involved objects and generates an isomorphism indicating the most coherent analogy between them.

These partial distance functions compare objects at different levels of detail and have a complementary role in identifying analogies. Their results are aggregated into an overall distance measure by an empirically plausible functional form. This overall distance is transformed into a similarity measure which reflects the aptness of the identified analogies.

3.3.1 Distance Over the Identification of Objects

In order to distinguish between identical and non identical objects, we introduce a metric over the identification of objects, defined as:

Definition 3.20 : *The identification distance d_1 between two user objects, identified by the object identifiers $\#i$, $\#j$, is defined as a function:*

$$d_1 : \left(\bigcup_{i=0}^{\infty} (IUC_i \times IUC_i) \right) \cup \left(\bigcup_{i=0}^{\infty} (AUC_i \times AUC_i) \right) \rightarrow [0, \dots, 1]$$

such that

$$d_1(\#i, \#j) = \begin{cases} 1 & \text{if } \#i \neq \#j \\ 0 & \text{if } \#i = \#j \end{cases}$$

This definition realizes the principle of the ontological uniformity since it restricts d_1 only between objects of the same instantiation level, which are both individuals or attributes.

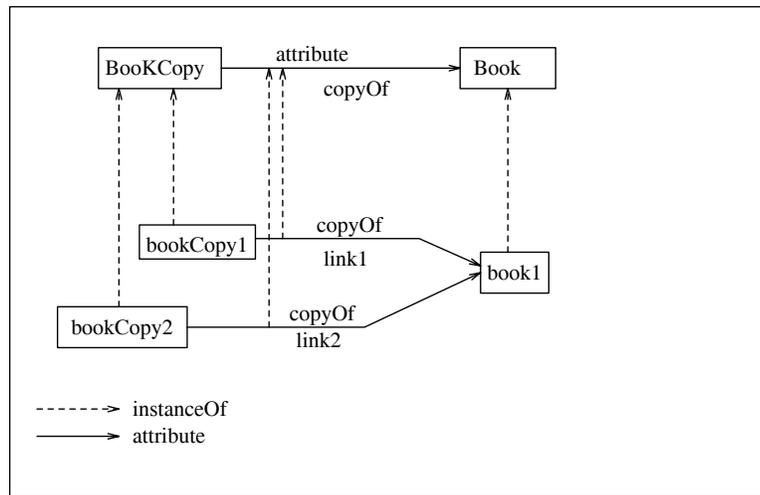


Figure 3.1: Non Identical Objects with Identical Semantics

This function allows a very coarse grain distinction between objects, but nevertheless it is useful in distinguishing between non identical objects with identical semantics. An example of such objects is presented in figure 3.1. The attributes *link1* and *link2* of objects *bookCopy1* and *bookCopy2* have identical semantics (i.e. they are both instances of the attribute class *copyOf* and have the same value *book1*) but are not identical.

3.3.2 Distance Over the Classification of Objects

The absolute distance over the classification of objects is measured according to the following function:

Definition 3.21 : The absolute classification distance D_2 between two user objects, identified by the object identifiers $\#i$, $\#j$, is defined as a function:

$$D_2 : \left(\bigcup_{k=0}^{\infty} (IUC_i \times IUC_i) \right) \cup \left(\bigcup_{k=0}^{\infty} (AUC_i \times AUC_i) \right) \rightarrow [0, \dots, \infty)$$

such that

$$D_2(\#i, \#j) = \sum_{x \in SDC[\#i, \#j]} \frac{1}{SD(x)}$$

where

$$SDC[\#i, \#j] = \left\{ o(\#i).In-o(\#j).In \cup o(\#j).In-o(\#i).In \right\}$$

This definition realizes the principle of ontological uniformity since it restricts D_2 only between objects of the same instantiation level, which are both individuals or attributes.

Function D_2 weights each non common class of two objects with the inverse of its depth in the Isa graph of its own instantiation level. These depths, that will be called *specialization depths* are estimated as the maximum length of the paths connecting each class with its *instantiation level superclass*. Formally, each class is associated with its instantiation level superclass, by the mapping obj_{\max} , which is defined as:

Definition 3.22 : obj_{\max} is a mapping defined as:

$$obj_{\max} : EXT[\#C] \rightarrow EXT[\#C] \text{ and}$$

$$obj_{\max}(\#i) = \#j \iff (\#j \in o(\#i).Isa) \text{ and } (o(\#j) \in B_i)$$

$obj_{\max}(\#i)$ will be called the *Instantiation Level superclass* of $\#i$.

As the following theorem indicates, each user class, has a unique Instantiation Level superclass:

Theorem 3.1 : obj_{\max} is an *M:1* mapping

Furthermore, if the set of the unordered superclasses of a class $\#i$ is defined as:

Definition 3.23 : For each user class, with identifier $\#i$, the set of its unordered superclasses, $US[\#i]$, is defined as:

$$US[\#i] = \left\{ x_1 \mid (x_1 \in o(\#i).Isa) \text{ and } (not(\exists x_2: (x_2 \in o(\#i).Isa) \text{ and } (x_1 \in o(x_2).Isa))) \right\}$$

The generalization graph having as root the class $\#i$ is defined as the set of its Isa relations according to the following definition:

Definition 3.24 : For each user class with identifier $\#i$, the set of its generalization relations, $GR[\#i]$, is defined as:

$$GR[\#i] = \left\{ (x_1, x_2) \mid (x_1 \in (o(\#i).Isa \cup \{\#i\})) \text{ and } (x_2 \in US[x_1]) \right\}$$

and the paths connecting #i with its instantiation level superclass form a set defined as:

Definition 3.25 : For each user class with identifier #i, the set of paths to its Instantiation Level superclass, $P[\#i]$, is defined as:

$$P[\#i] = \left\{ p \mid (p \subseteq GR[\#i]) \text{ and } (\forall x_1, x_2 : ((x_1, x_2) \in p) \Rightarrow ((x_1 = \#i) \text{ or } (x_2 = obj_{\max}(\#i)) \text{ or } (\exists x_3, x_4 : ((x_3, x_1) \in p) \text{ and } ((x_2, x_4) \in p)))) \right\}$$

Then, the specialization depth of #i, is defined as:

Definition 3.26 : The specialization depth $SD(\#i)$ of an object with identifier #i is defined as a function

$$SD : EXT[\#C] \rightarrow \{1, 2, 3, \dots\}$$

such that

$$SD(\#i) = \max_{p \in P[\#i]} (|p| + 1)$$

Since a class can have more than one superclasses, it might be placed at different specialization depths, considering distinct Isa hierarchies. To overcome this ambiguity, the specialization depth of a class is defined as the maximum of those depths and thus, it gives an absolute indication of the specificity of a class.

Thus, according to the definition of D_2 , the more specific a non common class is, the less its contribution to the classification distance of the objects under comparison. This is because the more specific classes, tend to express fine grain distinctions between homogeneous populations of objects, while the more general ones tend to express basic partitions between objects of different ontologies.

An example of how the absolute generalization distance is measured, is given in reference to figure 3.2. This figure presents a classification schema which groups classes of agents that interact with systems borrowing resources, such as a library or a car renting agency (i.e. the meta class *AgentClass*) and classes of entities, for which it is necessary to keep information, in order to support the borrowing activities of this system (i.e. the meta class *EntityClass*). The classes of entities are further distinguished into those for which

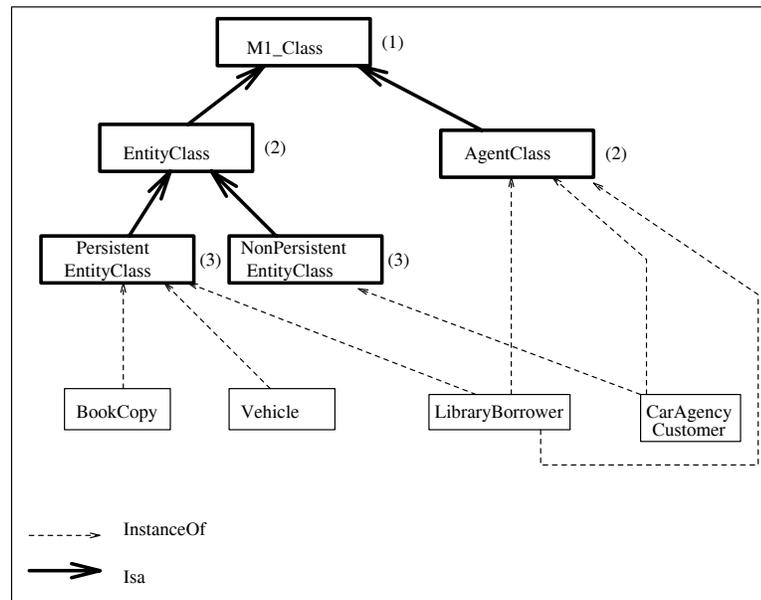


Figure 3.2: A Classification Schema for Entity and Agent Classes

the borrowing system must permanently keep information (i.e. the meta class *PersistentEntityClass*) and those entity classes, for which no information must be kept after the completion of the borrowing activities involving them (i.e. the meta class *NonPersistentEntityClass*). The numbers in parentheses are the specialization depths of the relevant classes. Given their classification under this meta schema, the absolute classification distance between the class representing the borrowers of a library system (i.e. class *LibraryBorrower*) and the class representing the customers of a car rental agency (i.e. the class *CarAgencyCustomer*) is 0.66. Also, the absolute classification distance between the rentable vehicles of the car agency (i.e. the class *Vehicle*) and the car agency customers is 1.16. Notice that, these latter classes resemble with each other in the sense that, they both reflect entities for which is necessary to keep information, since they are both instances of the meta class *EntityClass*. However, their absolute classification distance is greater than the distance of the former pair, because they differ in an important aspect: agenthood. The importance of this aspect is reflected by the specialization depth of the meta class *AgentClass* in the schema, which is 2.

The use of the specialization depth, as the weight of a class, in computing classification distances, can be further justified from a different perspective, related to the kinds of attributes which are introduced by some class depending on how general or specific it is. It can be reasonably assumed and, in fact it has been empirically verified [RMGJB76] that the deeper a class in an Isa hierarchy the less important the kinds of attributes it introduces. Subject to this hypothesis, the more general non common classes of two objects, will tend to introduce more important categories of attributes, than the less general ones. Consider also that, these categories will be the classes of the attribute objects associated with the instances of the non common classes. Hence, these non common classes will become the sources of attribute objects, with different semantics and importance, which will augment the differences between the compared instances having these attributes (see discussion about the semantic homogeneity of attributes below).

In addition to these characteristics, function D_2 has the basic properties of pseudometrics, as the following theorem states:

Theorem 3.2: *The function D_2 is a pseudometric.*

As a pseudometric, D_2 takes only positive values, is symmetric and triangular but it cannot distinguish between identical and non identical objects. For instance, the D_2 distance, between the classes *BookCopy* and *Vehicle* in figure 3.2, equals the D_2 distance between the class *BookCopy* and itself (i.e. 0), although the former pair of classes are not identical.

According to the principle of normalization, absolute measures D_2 are normalized into relative ones by the following homographic transformation.

Definition 3.27 : *The normalized classification distance d_2 between two user objects, identified by the object identifiers $\#i$, $\#j$, is defined as a function:*

$$d_2(\#i, \#j) = \frac{\beta_2 D_2(\#i, \#j)}{\beta_2 D_2(\#i, \#j) + 1}$$

The parameter β_2 in this definition, tunes the relative distance measures d_2 , since it determines the rate of the asymptotic convergence of these measures to 1, as D_2 goes to infinity. This is because the first derivative of the function d_2 , with respect to D_2 , is:

$$\frac{\partial d_2}{\partial D_2} = \frac{\beta_2}{(\beta_2 D_2 + 1)^2}$$

The parameter β_2 is set empirically, so that d_2 be very close to 0.5, when D_2 takes its average observed value.

As the following theorem verifies, the homographic normalization of absolute classification distance measures to relative ones, preserves the pseudometric properties of the former:

Theorem 3.3: *The function d_2 is a pseudometric.*

3.3.3 Distance Over the Generalization of Objects

The absolute generalization distance D_3 between objects is defined similarly to D_2 , as follows:

Definition 3.28 : *The absolute generalization distance D_3 , between two user objects, identified by $\#i$, $\#j$ is defined as a function:*

$$D_3 : (\bigcup_{i=0}^{\infty} (IUC_i \times IUC_i)) \rightarrow [0, \dots, \infty)$$

$$D_3(\#i, \#j) = \sum_{x \in SDSC[\#i, \#j]} \frac{1}{SD(x)}$$

where

$$SDSC[\#i, \#j] = \left\{ o(\#i).Isa \cup \left\{ \#i \right\} - o(\#j).Isa \cup \left\{ \#j \right\} \cup \left\{ o(\#j).Isa \cup \left\{ \#j \right\} - o(\#i).Isa \cup \left\{ \#i \right\} \right\} \right\}$$

This definition realizes the principle of ontological uniformity since it restricts D_3 only between objects of the same instantiation level, which are both individuals or attributes.

An example of absolute generalization distance measures is given in reference to figure 3.3. This figure presents a taxonomy of agent classes interacting with systems supporting the borrowing of resources.

According to this schema, the kind of the system which an agent interacts with (i.e. library vs. car rental agency) and the particular kind of this interaction (i.e. an agent may interact as an employee or as a borrower) are expressed by classes at equal depths (i.e. 3) and thus equal importance. Consequently, the absolute generalization distance between the classes *CarAgencyCustomer* and *CarAgencyEmployee* equals the absolute generalization distance between the classes *CarAgencyCustomer* and *LibraryBorrower*, which is 1.16 ($1/3 + 1/3 + 1/4 + 1/4$).

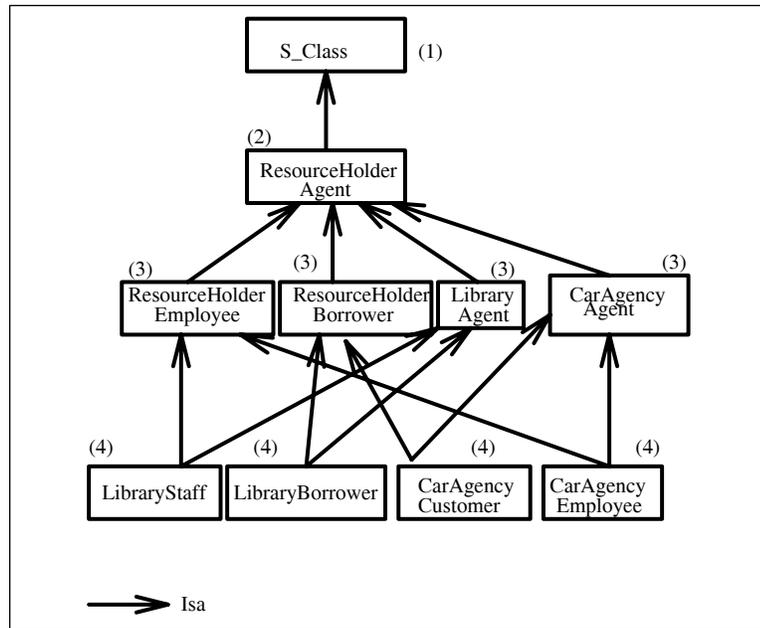


Figure 3.3: A Conceptual Schema of Agent Classes

D_3 is a metric as the following theorem indicates.

Theorem 3.4: *Function D_3 is a metric*

The absolute generalization distance measures are normalized according to the principle of normalization by a homographic transformation, as the following definition specifies:

Definition 3.29: *The normalized generalization distance d_3 between two user objects, identified by $\#i$, $\#j$, is defined as a function:*

$$d_3(\#i, \#j) = \begin{cases} \frac{\beta_3 D_3(\#i, \#j)}{1 + \beta_3 D_3(\#i, \#j)} & \text{if } \#i, \#j \in IUC_i \\ d_o(\#i, \#j) & \text{if } \#i, \#j \in AUC_i \end{cases}$$

where

$$d_o(\#i, \#j) = \begin{cases} 1 & \text{if } OC_{\#i} \neq OC_{\#j} \\ 0 & \text{if } OC_{\#i} = OC_{\#j} \end{cases}$$

According to this definition, the generalization distance over attribute classes, distinguishes between cases where the involved attribute classes have the same original class

and cases where they do not.

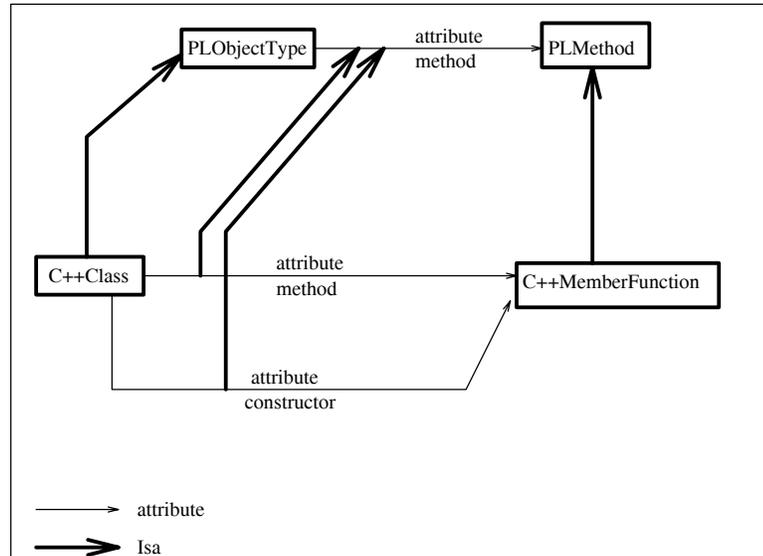


Figure 3.4: A Conceptual Schema for Object Types of OO Languages

In reference to figure 3.4, which presents a taxonomy of object types in object-oriented programming languages in the spirit of [Web92], the d_o distance between the attribute class *constructor* of the class *C++Class* and the attribute class *method* of the class *PLObjectType* is 1. This measure reflects the fact, that although the former of these attribute classes is a special case of the latter, it does not express the same relationship. In fact, the attribute class *constructor* represents the relationship between the special routines serving as constructors for C++ objects [Str88], while the attribute class *method* represents any relation between an object type and its routines.

On the other hand, the d_o distance between the attribute classes *method* of the classes *C++Class* and *PLObjectType* is 0, since these two attribute share the same original class (i.e. the attribute class *method* of the class *PLObjectType*). In fact, both these attribute classes express the same general relationship between object types and routines, as it is specialized by object types, in different programming languages.

The normalized measure d_3 preserves the metric properties of function D_3 , as the following theorem states:

Theorem 3.5: *Function d_3 is a metric.*

3.3.4 Distance Over the Attribution of Objects

As it may be apparent from the sections describing the modeling framework of similarity analysis, the abstraction of attribution is *semantically overloaded*[HK87a]. In other words, it may be used for describing very different aspects of objects, such as their structural decomposition, non structural relations with other objects or even simple properties. An attempt at comparing attributes of such diverse substance, would inevitably lead to uninterpretable results. As an extreme case, consider a comparison between the price of a car, that is to be borrowed by a car rental agency and the author of a book, that is to be lent by a library. On the other hand, it would make sense to compare the transportation number of a car, with the library code of a book, since both these attributes identify the relevant objects as resources to be borrowed.

Therefore, it is necessary to adopt criteria for determining whether or not two attributes have common semantics and as such they could be compared (i.e. mapping criteria). There are two basic approaches to defining such criteria.

The first assumes that conceptual models are built around a predefined and fixed set of attributes, which covers all the known interobject relationships in a certain domain(i.e. the *uniform representation assumption*). Given this perspective, only identical attributes may be assumed as having common semantics and thus be comparable with each other. This is the usual approach in case-based reasoning systems but it is also adopted in some models of analogical reasoning[Gen88a, Grein88b, FFG90], as discussed in chapter 2. However, it must be pointed out that unless the domains of interest are characterized by well defined relations and mature conceptual models, the *uniform representation assumption* could be reasonably questioned. Also, it is hardly acceptable for interdomain comparisons, where fixed feature sets, even if they exist for each single domain, are likely to be different for different domains. Furthermore, this approach neglects the possibility that non identically named attributes, such as the attributes *constructor* and *method* in figure 3.4, may share common semantics. Thus, the uniform representation assumption cannot lead to acceptable criteria for valid comparisons in analogical reasoning, in general.

The second approach is to identify criteria of attribute equivalence[BK88], indicating cases of non identical attributes that could be compared with each other. This approach is usually adopted in analogical reasoning[Win80,BC85, Thag88,THNG90].

Semantic Homogeneity of Attributes. In this thesis, we adopt the second approach due to our interest in devising a similarity model adequate for both intra and inter domain analogical reasoning. We introduce the relation of *semantic homogeneity* as a semantic equivalence relation, determining whether or not pairs of attributes can be compared with and mapped onto each other. Formally, this relation is defined as follows:

Definition 3.30: *Two attribute objects, identified by the object identifiers #i , #j, are semantically homogeneous sh(#i,#j), if and only if:*

$$OCL[\#i] = OCL[\#j]$$

where

$$OCL[x] = \left\{ x_1 \mid (x_1 = OC_{x_2}) \text{ and } (x_2 \in o(x).In) \right\} \quad x = \#i, \#j$$

In words, this definition requires two attributes to be classified under exactly the same original attribute classes for being semantically homogeneous.

In reference to case A of figure 3.5, the attributes *libraryCode* of the class *BookCopy* and *transportationNum* of the class *Vehicle* are semantically homogeneous since they are both classified under the attribute metaclass *identifiedBy*. This meta class groups all the attributes serving as identifiers of entity classes. It is exactly this unique identification semantics which justifies the analogy between these attributes.

Notice that, the equality of the original classes is necessary for allowing attributes belonging to different subclasses of the same original class to be considered as candidates for mapping. For example, the transportation numbers of specific vehicles, such as a car and a motorbike, classified under two distinct subclasses of the class *Vehicle* of figure 3.5, namely the classes *Car* and *Motorbike*, could be mapped onto each other even if these classes had specialized the attribute *transportationNum* inherited from the class *Vehicle*. This would be allowed since such specialized classes of attributes would share the same original class (i.e. the attribute class *transportationNum* of the class *Vehicle*).

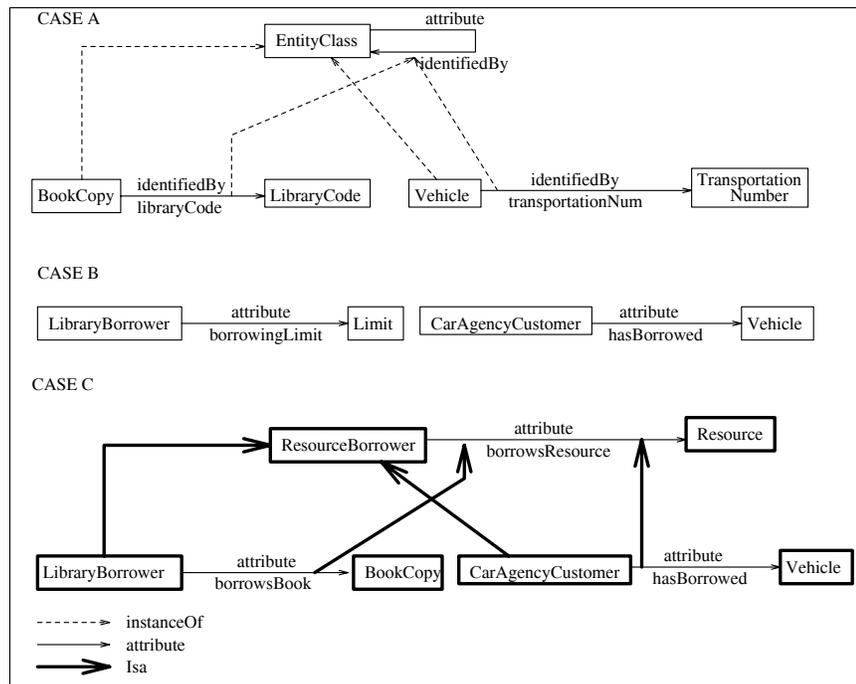


Figure 3.5: Semantic Overloading and Homogeneity of Attributes

As the following theorem states, semantic homogeneity is a reflexive, symmetric and transitive relation.

Theorem 3.6: *Semantic homogeneity of attributes is an equivalence relation*

Thus, it not only allows comparisons that would be possible under the *uniform representation* approach (reflexivity) and makes comparisons associative(symmetry), but it also makes sure that if an attribute x of an object i is comparable with an attribute y of an object j , which in turn is comparable with an attribute z of an object l , then x will also be comparable with z . This transitivity is a critical condition for defining a metric over the attribution of objects, as it will be evident in the proof of the *theorem 3.8*.

Semantic homogeneity extends the usual restriction of analogical comparisons only between constructs having an identical type [Win80], for the representation framework of our model, where any object and thus attribute objects may have one or more classes. As discussed in chapter 2, common classification implies that the relevant attributes have the same substance(a similar perspective is adopted in [MBJK90], where attribute classes are

realized as grouping together attributes with semantically similar roles with respect to the object they describe). Thus, semantic homogeneity restricts valid comparisons only between attributes of the same substance.

However, it must be pointed out that subject to how general or specific classes of attributes are, semantic homogeneity may fail to overcome the problem of the semantic overloading of attribution. This will be likely in cases where conceptual models are built without using a customized meta model attaching semantics to their elements. Such an example is presented in case B of figure 3.5. In this case, the attribute *borrowingLimit* of the class *LibraryBorrower* (i.e. the maximal number of books that a borrower may borrow from a library) would be considered as semantically homogeneous to the attribute *hasBorrowed* of the class *CarAgencyCustomer* (i.e. the car(s) that a customer has rent from a car rental agency), since they are both instances of the attribute class *Attribute*.

A possible solution to this problem would be to revise the definition of semantic homogeneity by requiring the existence and the identity of the original class of only user defined attribute classes (not built-in, such as *Attribute*), for considering two attributes as being homogeneous. However, this alternative could prove to be very restrictive. This is because conceptual models which are not built according to some meta model, may abstract analogical aspects along the generalization abstraction. Such an example, is illustrated in the case C of figure 3.5. In this case, the attribute classes *borrowBook* of the class *LibraryBorrower* and *hasBorrowed* of the class *CarAgencyCustomer* are both subclasses of the attribute class *borrowResource* of the class *ResourceBorrower*. Thus, they are analogous in the sense that they both express a relationship between a borrower and a borrowed resource. However, they are only instances of the built-in class *Attribute* and they are not classified under any user defined class. Notice that the identity of the superclasses, instead of the classes, cannot provide a general criterion of semantic homogeneity, since the generalization abstraction is not applicable to atomic objects (i.e. Tokens).

Another, possible definition of semantic homogeneity requiring either the identity of all the classes or all the superclasses, would make it a non equivalence relation, with obvious effects to the transitivity of the acceptable comparisons.

In summary, if a conceptual model comprises a meta model abstracting the basic types of relations in it, semantic homogeneity deals effectively with the problem of semantic overloading of attribution. In other cases, it is less powerful in resolving the problem of semantic overloading, but it is, nevertheless, tolerant to the expression of common semantics using the generalization abstraction.

On the basis of semantic homogeneity, we can define the set of *semantically homogeneous* pairs of attributes of two objects, as follows:

Definition 3.31: *The set of the semantically homogeneous of two user objects $SH[\#i, \#j]$, identified by the object identifiers $\#i$, $\#j$, is defined as:*

$$SH[\#i, \#j] = \left\{ (x_1, x_2) \mid (x_1 \in INT[\#i]) \text{ and } (x_2 \in INT[\#j]) \text{ and } sh(x_1, x_2) \right\}$$

The set $SH[\#i, \#j]$ includes all the possible partial analogies that may exist between the attributes of objects $\#i$ and $\#j$, and provide the ingredients for aggregating a global analogy between them.

Such a global analogy in general may be expressed as a relation R of the following form:

$$R : INT[\#i] \times INT[\#j]$$

R may have different properties, as evidenced from our survey of computational models of analogical reasoning, in chapter 2. A basic source of differentiation is whether R is an isomorphism as in[Gen88a,FFG90] or an M:1 mapping as in[BC85,Ind86,THING90] (see table 2.2 in chapter 2, for a summary of the solutions of computational models of analogical reasoning with respect to this aspect). Notice that isomorphisms are invertible into isomorphisms, while the inverse of an M:1 mapping may not be an M:1 mapping itself. This invertibility complies with our basic intuition that an analogy between two objects must be grounded on exactly the same object elements, regardless of which is the source and the target of the analogy(i.e. symmetry), something that wouldn't be the case with M:1 mappings. Similar arguments may be also be found elsewhere[Gen83], while evidence has been reported that humans interpret analogies in terms of isomorphic mappings[Gen88a].

Thus, the possible global analogies between objects are defined as the set of the possible isomorphisms between their semantically homogeneous attributes, according to the following definition:

Definition 3.32: *The set of the possible isomorphisms between the semantically homogeneous attributes of two user objects, identified by the object identifiers #i , #j, is defined as:*

$$C[\#i,\#j] = \left\{ c_k \mid (c_k \subseteq SH[\#i,\#j]) \text{ and } (\forall x_1, x_2 : \right. \\ \left. ((x_1, x_2) \in c_k) \Rightarrow (\text{not } (\exists x_3, x_4 : ((x_3, x_4) \in c_k) \text{ and } ((x_1 = x_3) \text{ and } (x_2 \neq x_4)) \text{ or } \\ ((x_1 \neq x_3) \text{ and } (x_2 = x_3)))))) \text{ and } (\text{not } (\exists c_{k'} : (c_{k'} \in C[\#i,\#j]) \text{ and } (c_{k'} \subset c_k))) \right\}$$

The set $C[\#i,\#j]$ may be thought of as the set of all the possible interpretations of the analogy between the objects #i and #j.

Given any such possible interpretation, the non analogous attributes of the involved objects are defined as:

Definition 3.33: *The set of the non analogous attributes of a user object #i, with respect to the object #j, given their isomorphism c_k , is defined as:*

$$A_{\#i}[c_k] = \left\{ x_1 \mid (x_1 \in INT[\#i]) \text{ and } (\text{not } (\exists x_2, x_3 : ((x_2, x_3) \in c_k) \text{ and } (x_1 = x_2))) \right\}$$

In words, the non analogous attributes of an object #i(#j), given a global analogy c_k between itself and another object #j(#i), are those attributes of it that map onto no attribute of #j(#i), under c_k .

The attributes of object #i, which are not semantically homogeneous to any attributes of object #j are also non analogous to any attribute of #j, under any possible interpretation of the analogy between #i and #j:

Theorem 3.7: *The attributes of an object #i, which are not semantically homogeneous to any attribute of an object #j belong to $A_{\#i}[c_k]$, for all c_k in $C[\#i,\#j]$*

However, the attributes of an object which are semantically homogeneous to some attribute(s) of another object, may not be considered as being analogous to them, under certain interpretations of an analogy.

In summary, the non analogous attributes have a fixed and a variant part. The fixed part depends on the criterion of the semantic homogeneity, since the semantically

heterogeneous attributes will anyway belong to it. On the other hand, the variant part depends on the selection of the particular interpretation of the analogy.

The Distance Function . Given all the possible interpretations of an analogy between two objects, the distance function over the attribution of objects is introduced to select the most plausible among them. This selection is based on a plausible optimality criterion, demanding that the chosen isomorphism must have the minimal total distance, as it is measured by adding the distance measures between all the individual pairs of attributes that constitute it. These measures are obtained by estimating the distance of each pair of attributes with respect to their classification, generalization and attribution as well as with respect to their values. Thus, measuring the distance over attribution requires a recursive analysis of the analogies not only between the attributes of the involved attributes but also between the values of these attributes.

Formally, the absolute attribution distance between two objects is defined as:

Definition 3.34 : *The absolute attribution distance D_4 , between two user objects, identified by $\#i$, $\#j$, is defined as a function:*

$$D_4 : \left(\bigcup_{k=0}^{\infty} (IUC_i \times IUC_i) \right) \cup \left(\bigcup_{k=0}^{\infty} (AUC_i \times AUC_i) \right) \times \text{PowersetOf} (IU \cup AU) \rightarrow [0, \dots, \infty)$$

such that

$$D_4(\#i, \#j, V) = \begin{cases} \infty & \text{if } INT[\#i] = \emptyset \text{ or } INT[\#j] = \emptyset \\ \min_{c_k \in C[\#i, \#j]} \left\{ \sum_{(x_1, x_2) \in c_k} s(x_1) s(x_2) d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 \right\} & \text{otherwise} \end{cases}$$

where

$$d'(\#i, \#j, V) = \frac{\beta D'(\#i, \#j, V)}{\beta D'(\#i, \#j, V) + 1}$$

$$D'(\#i, \#j, V) = \begin{cases} (\underline{PD}_{ec} W_1 \underline{PD}_{ec}^T)^{1/2} & \text{if } (\#i, \#j \in \bigcup_{k=1}^{\infty} AUC_i) \text{ and } ((o(\#i).TO \in V) \text{ or } (o(\#j).TO \in V)) \\ (\underline{PD}_{et} W_2 \underline{PD}_{et}^T)^{1/2} & \text{if } \#i, \#j \in AUC_0 \text{ and } ((o(\#i).TO \in V) \text{ or } (o(\#j).TO \in V)) \\ D(\#i, \#j, V) & \text{otherwise} \end{cases}$$

$$W_1 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 6 & 6 & 6 \\ 0 & 1 & 6 & 1 \end{bmatrix} \quad W_2 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 6 \\ 0 & 6 & 6 \end{bmatrix}$$

$$\underline{PD}_{ec} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_3(\#i, \#j) \quad d_4(\#i, \#j)]$$

$$PD_{et} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_4(\#i, \#j, V \cup \left\{ \#i, \#j \right\})]$$

V is a set of identifiers of objects ($V \subseteq IU \cup AU$) and,

$$s(x_i) \in [0, \dots, 1], \quad s(x_i) = \max_{x_j \in OCL[x_i]} \left\{ SL(x_j) \right\} \quad SL(x_j) \text{ is the salience of } x_j$$

This definition realizes the principle of ontological uniformity since it restricts D_4 only between objects of the same instantiation level, which are both individuals or attributes.

According to this definition, an isomorphism c_k is selected as the optimal interpretation of an analogy, if it minimizes the measure:

$$\sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#}[c_k]} s(x_4)^2$$

Thus, it may be understood as the most coherent interpretation of the analogy between the involved objects, grounded on further coherent interpretations of analogies between other objects, connected with them, along the entire transitive closures of their attributions (see figure 3.6).

The factors $s(x_i)$, are weights expressing how important an attribute x_i is for its possessing object. These weights, as indicated by definition 3.34, are estimated as the maximum measure SL for each of the original classes of the attribute. SL refers to the *salience* of the attribute classes and its exact evaluation is described in the next chapter. However, it is important to point out that for each pair of attributes (x_1, x_2) , which belong to an isomorphism c_k , $s(x_1)$ will be equal to $s(x_2)$, since

$$OCL[x_1] = OCL[x_2]$$

This means, that semantically homogeneous attributes have the same importance for their possessing objects.

The set V in definition 3.34 determines the scope in which the attribution distance is evaluated. This set is introduced in order to avoid an infinite recursion that could be caused by the presence of *cyclic* attributes. These are attributes, whose values are objects, the distance of which to some other object is currently being estimated.

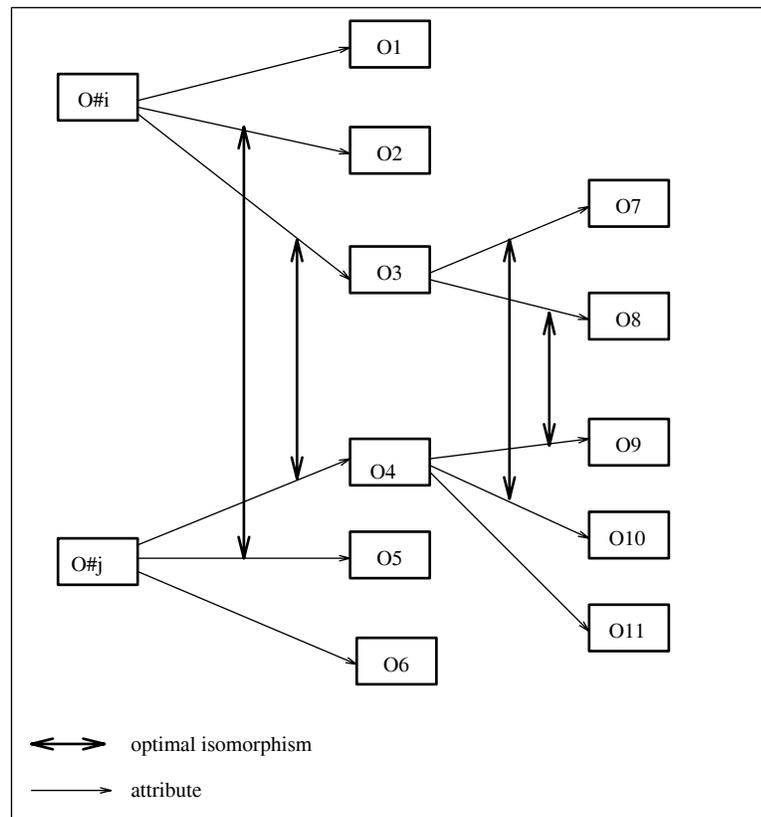


Figure 3.6: Synthesis of Successive Coherent Interpretations of Analogies

Figure 3.7 presents an example of such attributes. The estimation of the attribution distance between the class *LibraryBorrower* and the class *VideoClubCustomer* in this figure would require the estimation of the overall distance between the attributes *borrows* of the former and the latter class. The estimation of this overall distance would itself require the estimation of the distance between the classes *LibraryBorrower* and *VideoClubCustomer*. In such cases, recursion is controlled by estimating the distance between the pairs of attributes that caused the cycles (i.e. the attributes *hasEnrolledBorrower* and *hasCustomer*) only with respect to their identification, classification, generalization and attribution but not with respect to their values.

This distinction is introduced by function D' , which computes the overall pairwise distances of attributes in a specific scope of similarity analysis. The matrices determining the quadric form of function D' correspond to empirically verified correlations between

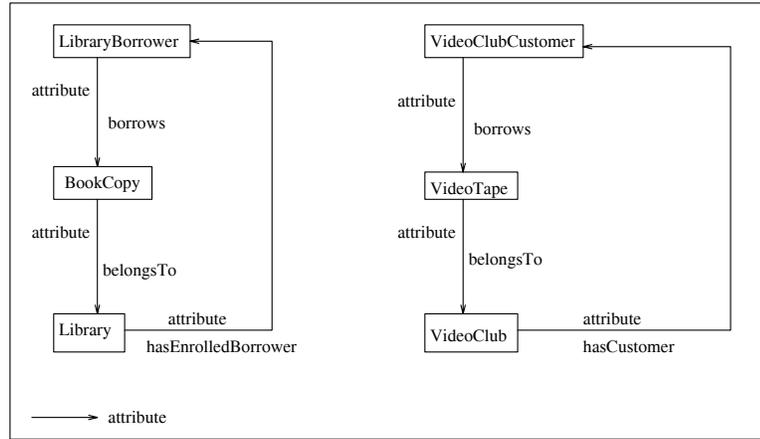


Figure 3.7: Cyclic Attributes

partial distances(see section 3.4 below). Furthermore, the elements of matrix w_1 , which are the multiplying coefficients for the generalization distance (i.e. matrix elements equal to 6), have been selected so as to increase the overall distance between attributes, whose generalization distance equals to 1 and, consequently to disfavor mappings between such attributes(see *theorem 3.10* and its proof in the mathematical appendix of this chapter).

The definition of function D_4 adopts a conservative approach by assigning a distance equal to infinity to pairs of objects, whose intensions are empty. This approach is conservative because it assumes that if the relevant intensions were not empty, it would be more likely for them to include semantically dissimilar attributes rather than semantically similar ones.

D_4 is symmetric, triangular, can distinguish between identical and non identical objects and takes positive values, according to the following theorem:

Theorem 3.8: *Function D_4 is a metric.*

By the principle of normalization, the absolute distance measures D_4 , are normalized into relative measures, according to the following definition:

Definition 3.35: *The normalized attribution distance d_4 between two user objects, identified by the object identifiers $\#i$, $\#j$, is defined as a function:*

$$d_4(\#i, \#j) = \frac{\beta_4 D_4(\#i, \#j)}{\beta_4 D_4(\#i, \#j) + 1}$$

The parameter β_4 in this definition, tunes the relative distance measures, since it determines the rate of the asymptotic convergence of these measures to 1, as D_4 goes to infinity.

As the following theorem indicates, the homographic transformation of definition 3.35, preserves the metric properties:

Theorem 3.9: *Function d_4 is a metric function .*

Another important implication of definition 3.34 is that in cases where one or more pairs of semantically homogeneous attributes of two objects under comparison have the same original class, these pairs will necessarily be elements of the optimal isomorphism between these objects, according to the following theorem:

Theorem 3.10: *All the pairs of attribute objects (x_1, x_2) of two objects $\#i, \#j$ that belong to the set $SH[\#i, \#j]$ and have the same original class will belong to the optimal isomorphism c_{opt} between objects $\#i$ and $\#j$.*

This theorem is a consequence of the particular quadric forms of the distance function D' , and the definition of the generalization distance d_3 between attribute classes. The tendency to select isomorphisms, which map attributes with the same original class, is reasonable considering that a common original class, indicates that the relevant attributes express the same relation, as already discussed.

An example of such a case is presented in figure 3.8. In reference to this figure, both attribute classes of the class *LibraryBorrower* are semantically homogeneous to both attribute classes of the class *VideoClubCustomer* and therefore they may be mapped to any of them. However, the generalization distance between the attribute classes *borrowResource* of the classes *LibraryBorrower* and *VideoClubCustomer* is 0. These attribute classes, although non identical themselves(i.e. they are attribute objects with different identifiers) express the same relationship, between a borrower and the resource (s)he borrows, for different kinds of borrowers(i.e. library borrowers and customers of video clubs) and resources(i.e. copies of books and video tapes).

On the other hand, the generalization distances between each of the attribute classes *hasRequested* and *prefers* and the attribute classes *borrowResource* of the classes *LibraryBorrower* and *VideoClubCustomer* are all equal to 1. Hence, assuming that all

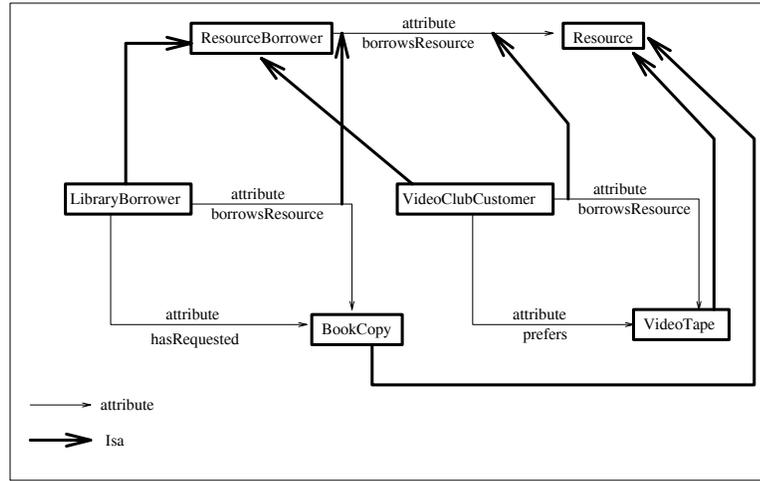


Figure 3.8: Attributes with an Identical Original Class

the other partial distances between the possible pairs of attributes of the involved entity classes, are equal, the final optimal isomorphism, maps the attribute classes *borrowsResource* of the classes *LibraryBorrower* and *VideoClubCustomer* onto each other.

3.4 The Overall Distance Function

The partial metrics d_1, d_2, d_3 and d_4 are aggregated into an overall distance D according to the following quadric function:

Definition 3.36 : *The overall distance D , between two user objects, identified by $\#i, \#j$, is defined as a function:*

$$D : \left(\bigcup_{k=0}^{\infty} (IUC_i \times IUC_i) \right) \cup \left(\bigcup_{k=0}^{\infty} (AUC_i \times AUC_i) \right) \times \text{powersetOf} (IU \cup AU) \rightarrow [0, \dots, \infty)$$

such that

$$D(\#i, \#j, V) = \begin{cases} (\underline{PD}_{ec} \underline{W}_{ec} \underline{PD}_{ec}^T)^{1/2} & \text{if } \#i, \#j \in \bigcup_{k=1}^{\infty} IUC_k \\ (\underline{PD}_{et} \underline{W}_{et} \underline{PD}_{et}^T)^{1/2} & \text{if } \#i, \#j \in IUC_0 \\ (\underline{PD}_{ac} \underline{W}_{ac} \underline{PD}_{ac}^T)^{1/2} & \text{if } \#i, \#j \in \bigcup_{k=1}^{\infty} AUC_k \\ (\underline{PD}_{at} \underline{W}_{at} \underline{PD}_{at}^T)^{1/2} & \text{if } \#i, \#j \in AUC_0 \end{cases}$$

where

$$PD_{ec} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_3(\#i, \#j) \quad d_4(\#i, \#j)]$$

$$PD_{et} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_4(\#i, \#j, V \cup \{\#i, \#j\})]$$

$$PD_{ac} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_3(\#i, \#j) \quad d_4(\#i, \#j, V \cup \{\#i, \#j\}) \quad D(o(\#i).TO, o(\#j).TO, V \cup \{\#i, \#j\})]$$

$$PD_{at} = [d_1(\#i, \#j) \quad d_2(\#i, \#j) \quad d_4(\#i, \#j, V \cup \{\#i, \#j\}) \quad D(o(\#i).TO, o(\#j).TO, V \cup \{\#i, \#j\})]$$

$$\underline{W}_{ec} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix} \quad \underline{W}_{et} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\underline{W}_{ac} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 6 & 6 & 6 & 0 \\ 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \underline{W}_{at} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Notice that D is a recursive function conforming to the principle of ontological uniformity.

A series of experiments showing a statistically significant correlation between the classification(d_2), generalization(d_3) and the attribution(d_4) distance measures prompted the use of a quadric functional form for the overall distance D.

Table 3.1 presents measured correlation coefficients in 10 experiments over object bases. These object bases included conceptual schemas and populations of requirements specifications(cases 1,2,3,4), abstractions for specifying the domain aspects of information systems as they are defined in[MS92](case 5), static analysis data of CoolL code modules[Web92], a model describing constructs of the C++ programming language[DK93](cases 7,8) and a model for describing museum artifacts[CCD92] (cases 9,10).

As shown in this table, in 4 out of the 6 cases where both d_2 and d_3 were definable(i.e. the involved objects were not tokens) and not equal to 0, there was found a positive and statistically significant correlation coefficient between their resulting measures. Coefficients marked with single(*) and double(**) asterisks were found significant at the levels 0.01 and 0.05, respectively. Non marked coefficients were not statistically significant. Also, in all the 7 cases of object sets for which d_2 distances were not equal to 0, these distances

had a positive correlation with the attribution distances. These coefficients were statistically significant in 4 cases.

Case	Set Size	d_2, d_3	d_2, d_4	d_3, d_4
1	231	-	-	.81*
2	28	.47**	.59*	.6*
3	6	0	.74	.08
4	861	.45*	.12*	.51*
5	10	-	-	.53
6	903	-	.15*	-
7	171	.24*	.09	.64*
8	66	-0.05	.17	.21
9	741	-	-	.39*
10	946	.55*	.6*	.4*

Finally, in all the 9 cases of object sets for which d_3 distances were definable (i.e. the involved objects were not tokens), the correlation coefficients between these and the attribution distances were positive and in 6 of them statistically significant.

According to the following theorem, the aggregate function D is a metric.

Theorem 3.11: *Function D is a metric .*

3.5 The Similarity Function

The overall distance measures between objects are transformed into similarity measures, according to the following function:

Definition 3.37 : *The similarity S between two user objects, identified by $\#i, \#j$, is defined as a function:*

$$S : \left(\bigcup_{k=0}^{\infty} (IUC_i \times IUC_i) \right) \cup \left(\bigcup_{k=0}^{\infty} (AUC_i \times AUC_i) \right) \times \text{powersetOf} (IU \cup AU) \rightarrow [0, \dots, 1]$$

$$S(\#i, \#j, V) = e^{-N * D(\#i, \#j, V)}, N \in (0, \dots, \infty)$$

This definition, like all the definitions of the distance functions, is consistent with the principle of ontological uniformity and furthermore normalizes the transformed distance measures.

Notice also that since the first derivative of function S with respect to D is:

$$-Ne^{-ND}$$

any change to the value of D will cause an exponentially larger change to the value of S, the magnitude of which will depend on the parameter N(i.e. it determines the sensitivity of the S measures to the D measures).

A similar behavior of human assessments about analogical similarities between objects, with respect to observed differences between them, has been empirically verified in [She84] and discussed in [Rus88].

Measures S indicate the aptness of the analogy between two objects.

3.6 An Example of Similarity Evaluation

In this section, we give an example of similarity analysis between the objects of figure 3.9. This figure, presents a conceptual model of resource borrowers of a library(i.e. the class *LibraryBorrower*) and customers of a car renting agency(i.e. the class *CarAgencyCustomer*). Also, the detailed partial and overall distances as well as the relevant similarity measures are presented in table 3.2.

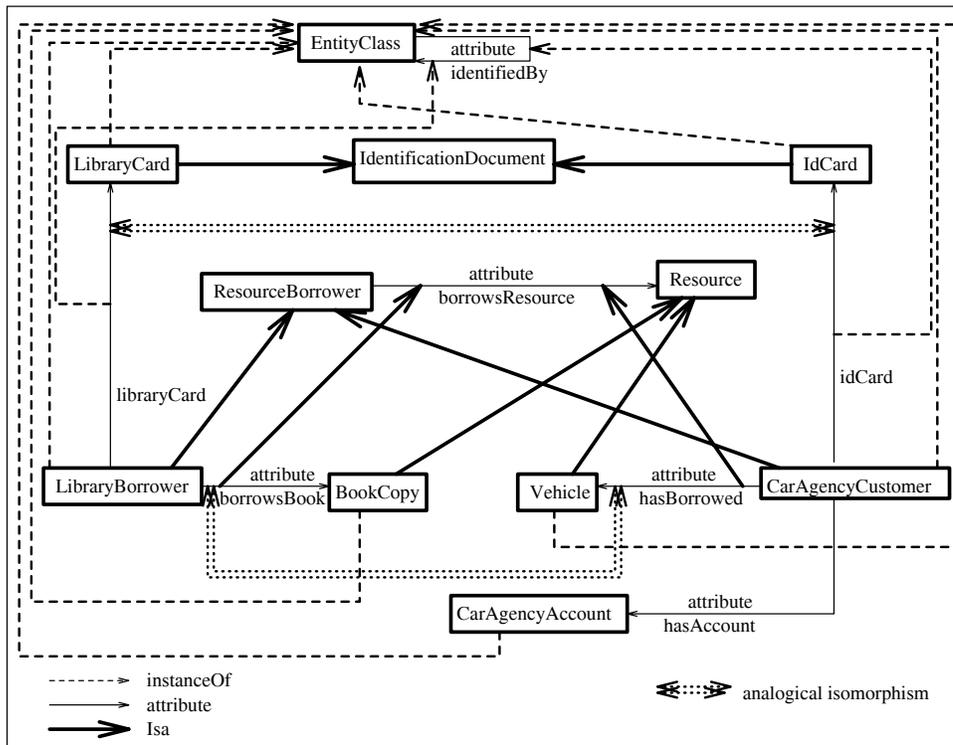


Figure 3.9: Analogy Between Library Borrowers and Car Agency Customers

As illustrated in figure 3.9, the analysis of similarity between these two objects, results in an optimal analogical isomorphism, mapping the attributes *libraryCard* and *borrowsBook* of the former class onto the attributes *idCard* and *hasBorrowed* of the latter, respectively. In fact, these attributes have common semantics. The first pair consists of attributes that enable the identification of the borrower to the resource borrowing system. The identification of the library borrowers is possible through a card awarded by the library, while the identification of the car agency customers is possible through their ordinary

identification cards. This dissimilarity does not devalidate the underlying analogy, which is acknowledged in the conceptual schema by the classification of the relevant attributes, under the attribute class *identifiedBy*. This analogy becomes more evident by the fact that both the possessing and the value objects of these attributes are generalized into common superclasses (i.e. the classes *ResourceBorrower* and *IdentificationDocument*).

Notice that, the optimal isomorphism also maps the attribute *borrowBook* of the class *LibraryBorrower* onto the attribute *hasBorrowed* of the class *CarAgencyCustomer*. It must be pointed out, that this was not the only mapping possibility considering the attribute *borrowBook* because, due to its semantic homogeneity with the attribute *hasAccount*, it could be mapped to this attribute, too.

Objects	d_1	d_2	d_3	d_4	D_{TO}	D	S
		$\beta_2 = 2$	$\beta_3 = 2$	$\beta_4 = .5$			$N=0.3$
LibraryBorrower,CarAgencyCustomer	1	0	0.56	0.34	-	1.61	0.61
BookCopy,Vehicle	1	0	0.56	1	-	1.36	0.66
LibraryCard,Vehicle	1	0	0.56	1	-	1.36	0.66
BookCopy,CarAgencyAccount	1	0	0.72	1	-	1.79	0.58
LibraryCard,CarAgencyAccount	1	0	0.72	1	-	1.79	0.58
borrowBook,hasBorrowed	1	0	1	1	1.36	2.2	0.51
libraryCard,idCard	1	0	1	1	1.36	2.2	0.51
borrowBook,hasAccount	1	0.5	1	1	1.79	2.33	0.49

However, as observed from table 3.2, which presents the pairwise distances between the objects involved in the analysis of similarity between the classes *LibraryBorrower* and *CarAgencyCustomer*, the overall distance of the pair of attributes (*borrowBook,hasBorrowed*) is less than the overall distance of the pair (*borrowBook,hasAccount*). This is due to the difference in the distance between the values of these

pairs. The final selected mapping of the attribute *borrowBook* onto the attribute *hasBorrowed* is also justified by the presence of a common generalization for these two classes of attributes (i.e. the attribute classes *borrowResource*).

3.7 Mathematical Appendix: Proofs of Theorems in Chapter 3

This appendix begins with a notation review and a formal definition of metrics and pseudometrics and then it turns to proving the theorems, that were presented in the preceding sections of the chapter.

Notation Preliminaries. The symbols $\forall \exists \Rightarrow \Leftarrow\Rightarrow$ and *or not* are used with their classical interpretation in logic. Object identifiers are denoted as symbols prefixed by the sign # and variables are denoted by the subscribed symbol x .

Metrics. Following [Gil87], a function d on a non empty set X , is a metric for X , if it assigns to each pair of elements of X , a real number (i.e. $d : X \times X \rightarrow R$), which satisfies the following axioms:

$$D(x,y) \geq 0 \quad \alpha_1$$

$$D(x,y) \equiv D(y,x) \quad \alpha_2$$

$$D(x,z) \leq D(x,y) + D(y,z) \quad \alpha_3$$

$$D(x,y) = 0 \iff x=y \quad \alpha_4$$

A function d which satisfies only the first three of these axioms is called [Kow65], a *pseudometric*.

Theorem 3.1 : *obj_{max} is an M:1 mapping*

PROOF: Suppose that an object x_1 is mapped onto two different Instantiation Level classes x_2 and x_3 . Then, according to the definition of the obj_{max} mapping, it must be $x_2 \in B_i$ and $x_3 \in B_i$

Then if x_4 and x_5 are the Instantiation level classes of x_2 and x_3 , due to axiom A.3.12, it must hold that:

$$(x_4 \in o(x_1).In) \text{ and } (x_4 \in B_i) \text{ and } (x_5 \in o(x_1).In) \text{ and } (x_5 \in B_i) \quad (I)$$

Moreover, according to the definition 3.8, each instantiation level class is classified under a different instantiation level class. Therefore, it must also be that:

$$x_4 \neq x_5 \quad (II)$$

However, the conjunction of (I) and (II) violates the axiom A.3.2 which requires that any non Basic class, and therefore object x_1 , can only be classified under a unique Instantiation Level class. \square

Lemma 3.1 For any three sets A, B, D it holds that

$$(A-B) \cup (B-A) \subseteq ((A-D) \cup (D-A)) \cup ((D-B) \cup (B-D))$$

PROOF: Let A^c denote the complement of a set A .

Then, according to the set theory, we have that:

$$\begin{aligned} (A-B) \cup (B-A) &= (A \cap B^c) \cup (B \cap A^c) = (A \cup (B \cap A^c)) \cap (B^c \cup (B \cap A^c)) = \\ &= ((A \cup B) \cap (A \cup A^c)) \cap ((B^c \cup B) \cap (B^c \cup A^c)) = (A \cup B) \cap (B^c \cup A^c) \quad (I) \end{aligned}$$

Thus, by applying (I) twice, we have that:

$$\begin{aligned} ((A-D) \cup (D-A)) \cup ((D-B) \cup (B-D)) &= ((A \cup D) \cap (D^c \cup A^c)) \cup ((D \cup B) \cap (D^c \cup B^c)) = \\ &= ((A \cup D) \cup ((D \cup B) \cap (D^c \cup B^c))) \cap ((D^c \cup A^c) \cup ((D \cup B) \cap (D^c \cup B^c))) = \\ &= (A \cup D \cup B) \cap (A^c \cup D^c \cup B^c) \quad (II) \end{aligned}$$

However, since

$$\begin{aligned} ((A \cup B) \cap (A^c \cup B^c)) \cap ((A \cup D \cup B) \cap (A^c \cup D^c \cup B^c)) &= \\ ((A \cup B) \cap (A \cup B \cup D)) \cap ((A^c \cap B^c) \cap (A^c \cup B^c \cup D^c)) &= ((A \cup B) \cap (A^c \cup B^c)) \end{aligned}$$

we have that

$$((A \cup B) \cap (A^c \cup B^c)) \subseteq ((A \cup D \cup B) \cap (A^c \cup D^c \cup B^c))$$

and thus, due to (I) and (II)

$$(A-B) \cup (B-A) \subseteq ((A-D) \cup (D-A)) \cup ((D-B) \cup (B-D)) \quad \square$$

Lemma 3.2: The homographic transformation:

$$d(x_1, x_2) = \frac{aD(x_1, x_2)}{aD(x_1, x_2) + 1}, \quad a > 0$$

of a metric D is a metric

PROOF: a) Axiom α_1 is satisfied since:

$$d(x_1, x_2) = \frac{aD(x_1, x_2)}{aD(x_1, x_2) + 1} \geq 0 \quad \text{and} \quad d(x_1, x_1) = \frac{aD(x_1, x_1)}{aD(x_1, x_1) + 1} = 0$$

b) Axiom α_2 is satisfied since:

$$d(x_1, x_2) = \frac{aD(x_1, x_2)}{aD(x_1, x_2) + 1} = \frac{aD(x_2, x_1)}{aD(x_2, x_1) + 1} = d(x_2, x_1)$$

c) Axiom α_3 is satisfied since:

$$\begin{aligned}
d(x_1, x_2) \leq d(x_1, x_3) + d(x_3, x_2) &\iff \frac{aD(x_1, x_2)}{aD(x_1, x_2) + 1} \leq \frac{aD(x_1, x_3)}{aD(x_1, x_3) + 1} + \frac{aD(x_3, x_2)}{aD(x_3, x_2) + 1} \iff \\
&\frac{a^3D(x_1, x_2)D(x_1, x_3)D(x_3, x_2) + a^2D(x_1, x_2)D(x_1, x_3) + a^2D(x_1, x_2)D(x_3, x_2) + aD(x_1, x_2)}{(aD(x_1, x_2) + 1)(aD(x_1, x_3) + 1)(aD(x_3, x_2) + 1)} \leq \\
&\frac{a^3D(x_1, x_3)D(x_1, x_2)D(x_3, x_2) + a^2D(x_1, x_3)D(x_1, x_2) + a^2D(x_1, x_3)D(x_3, x_2) + aD(x_1, x_3)}{(aD(x_1, x_2) + 1)(aD(x_1, x_3) + 1)(aD(x_3, x_2) + 1)} + \\
&\frac{a^3D(x_2, x_3)D(x_1, x_2)D(x_1, x_3) + a^2D(x_2, x_3)D(x_1, x_2) + a^2D(x_2, x_3)D(x_3, x_2) + aD(x_2, x_3)}{(aD(x_1, x_2) + 1)(aD(x_1, x_3) + 1)(aD(x_3, x_2) + 1)} \iff \\
&\frac{a^3D(x_1, x_2)D(x_1, x_3)D(x_3, x_2) + aD(x_1, x_2)}{(aD(x_1, x_2) + 1)(aD(x_1, x_3) + 1)(aD(x_3, x_2) + 1)} \leq \\
&\frac{2a^3D(x_1, x_3)D(x_1, x_2)D(x_3, x_2) + a^2D(x_1, x_3)D(x_3, x_2) + a^2D(x_3, x_2)D(x_1, x_3) + aD(x_1, x_3) + aD(x_3, x_2)}{(aD(x_1, x_2) + 1)(aD(x_1, x_3) + 1)(aD(x_3, x_2) + 1)}
\end{aligned}$$

which is true.

d) Axiom α_4 is satisfied since: $d(x_1, x_2) = 0 \iff D(x_1, x_2) = 0 \Rightarrow x_1 \equiv x_2$

□

Theorem 3.2: *The function D_2 is a pseudometric.*

PROOF: a) Axiom α_1 is satisfied since for all x such that $x \in EXT[\#C]$ it holds that $SD(x) > 0$. Therefore,

$$D_2(\#i, \#j) = \sum_{x \in (o(\#i).In - o(\#j).In)} \frac{1}{SD(x)} + \sum_{x \in (o(\#j).In - o(\#i).In)} \frac{1}{SD(x)} \geq 0$$

b) Axiom α_2 is satisfied since:

$$\text{Let } S_{ij} = \left\{ o(\#i).In \cup o(\#j).In \right\} \text{ and } S_{ji} = \left\{ o(\#j).In - o(\#i).In \right\}$$

We have that,

$$D_2(\#i, \#j) = \sum_{x_1 \in S_{ij}} \frac{1}{SD(x_1)} + \sum_{x_2 \in S_{ji}} \frac{1}{SD(x_2)} = \sum_{x_2 \in S_{ji}} \frac{1}{SD(x_2)} + \sum_{x_1 \in S_{ij}} \frac{1}{SD(x_1)} = D_2(\#j, \#i)$$

c) Axiom α_3 is satisfied since:

Let,

$$S_i = o(\#i).In \quad S_j = o(\#j).In \quad S_l = o(\#l).In$$

$$S_1 = (S_i - S_j) \cup (S_j - S_i) \quad S_2 = (S_i - S_l) \cup (S_l - S_i) \quad S_3 = (S_l - S_j) \cup (S_j - S_l)$$

According to *lemma 3.1*, we have that $S_1 \subseteq S_2 \cup S_3$ (I)

Also, whenever $f(x) > 0$ for all x such that $x \in A \cup B$, it holds that:

$$\sum_{x \in A \cup B} f(x) = \sum_{x \in A} f(x) + \sum_{x \in B} f(x) - \sum_{x \in (A \cap B)} f(x) \leq \sum_{x \in A} f(x) + \sum_{x \in B} f(x) \quad (II)$$

$$\text{and } \sum_{x \in A} f(x) \leq \sum_{x \in A \cup B} f(x) \quad (III)$$

Thus, since $\frac{1}{SD(x)} > 0$ for any $x \in C_i, i \geq 1$, (II) and (III) imply that

$$\sum_{x \in S_1} \frac{1}{SD(x)} \leq \sum_{x \in (S_2 \cup S_3)} \frac{1}{SD(x)} \leq \sum_{x \in S_2} \frac{1}{SD(x)} + \sum_{x \in S_3} \frac{1}{SD(x)}$$

Thus, since D_2 satisfies the axioms α_1 , α_2 and α_3 it is a pseudometric.

□

Theorem 3.3: *The function d_2 is a pseudometric.*

PROOF: This theorem is an implication of *theorem 3.2* and *lemma 3.2*.

□

Theorem 3.4: *Function D_3 is a metric.*

PROOF: Let

$$S_{ij} = \left\{ o(\#i).Isa \cup \left\{ \#i \right\} - o(\#j).Isa \cup \left\{ \#j \right\} \right\} \quad S_{ji} = \left\{ o(\#j).Isa \cup \left\{ \#j \right\} - o(\#i).Isa \cup \left\{ \#i \right\} \right\}$$

$$S_{il} = \left\{ o(\#i).Isa \cup \left\{ \#i \right\} - o(\#l).Isa \cup \left\{ \#l \right\} \right\} \quad S_{lj} = \left\{ o(\#l).Isa \cup \left\{ \#l \right\} - o(\#j).Isa \cup \left\{ \#j \right\} \right\}$$

a) Axioms α_1 and α_4 are satisfied since, for any $\#i$, $\#j$ it holds that

$$D_3(\#i, \#j) = 0 \iff \#i = \#j \text{ and } D_3(\#i, \#j) > 0 \iff \#i \neq \#j$$

These two relations can be proven as it follows:

i) Considering the implication $D_3(\#i, \#j) = 0 \implies \#i = \#j$ we have:

$$D_3(\#i, \#j) = 0 \implies \sum_{x \in S_{ij}} \frac{1}{SD(x)} + \sum_{x \in S_{ji}} \frac{1}{SD(x)} = 0 \quad (I)$$

However, since for all x such that $x \in C_i$, $i \geq 1$ $SD(x) > 0$ (I) implies:

$$\sum_{x \in S_{ij}} \frac{1}{SD(x)} = 0 \implies S_{ij} = \emptyset \implies o(\#i).Isa \cup \left\{ \#i \right\} \subset o(\#j).Isa \cup \left\{ \#j \right\} \quad (II)$$

and

$$\sum_{x \in S_{ji}} \frac{1}{SD(x)} = 0 \Rightarrow S_{ji} = \emptyset \Rightarrow o(\#j).Isa \cup \{\#j\} \subset o(\#i).Isa \cup \{\#i\} \quad (III)$$

$$\text{Furthermore, (II) and (III) imply that } o(\#i).Isa \cup \{\#i\} = o(\#j).Isa \cup \{\#j\} \quad (IV)$$

However, (IV) cannot be true unless $\#i = \#j$, since if $\#i \neq \#j$ then (IV) implies that

$$(\#i \in o(\#j).Isa) \text{ and } (\#j \in o(\#i).Isa)$$

that contradicts with axiom A.3.10, which requires that isa relations must be acyclic.

Thus, it must be $\#i = \#j$

ii) The equivalence $D_3(\#i, \#j) = 0 \iff \#i = \#j$ is valid since:

$$\#i = \#j \Rightarrow o(\#i).Isa \cup \{\#i\} = o(\#j).Isa \cup \{\#j\} \Rightarrow S_{ij} = \emptyset \text{ and } S_{ji} = \emptyset$$

$$\text{Thus, } d_3(\#i, \#j) = \sum_{x \in S_{ij}} \frac{1}{SD(x)} = 0 \text{ and } \sum_{x \in S_{ji}} \frac{1}{SD(x)} = 0$$

iii) The implication $\#i \neq \#j \Rightarrow D_3(\#i, \#j) > 0$ is valid since:

Due to axiom A.3.10, we have that:

$$\#i \neq \#j \Rightarrow (\text{not}(\#i \in S_{ij}) \text{ and } (\#j \in S_{ij})) \iff (\text{not}(\#i \in S_{ij})) \text{ or } (\text{not}(\#j \in S_{ij}))$$

Therefore, $(\{\#i\} \subseteq S_{ij}) \text{ or } (\{\#j\} \subseteq S_{ji})$ and

$$\sum_{x \in S_{ij}} \frac{1}{SD(x)} > 0 \text{ or } \sum_{x \in S_{ji}} \frac{1}{SD(x)} > 0$$

Thus,

$$D_3(\#i, \#j) = \sum_{x \in S_{ij}} \frac{1}{SD(x)} + \sum_{x \in S_{ji}} \frac{1}{SD(x)} > 0$$

iv) The implication $D_3(\#i, \#j) > 0 \Rightarrow \#i \neq \#j$ is valid since:

$$D_3(\#i, \#j) > 0 \Rightarrow (\sum_{x \in S_{ij}} \frac{1}{SD(x)} > 0 \text{ or } \sum_{x \in S_{ji}} \frac{1}{SD(x)} > 0) \Rightarrow S_{ij} \neq \emptyset \text{ or } S_{ji} \neq \emptyset$$

However, since

$$\#i = \#j \iff o(\#i).Isa \cup \{\#i\} = o(\#j).Isa \cup \{\#j\} \iff S_{ij} = S_{ji} = \emptyset$$

we have $D_3(\#i, \#j) > 0 \Rightarrow \#i \neq \#j$

b) D_3 satisfies the axiom α_2 since:

$$D_3(\#i, \#j) = \sum_{x \in S_{ij}} \frac{1}{SD(x)} + \sum_{x \in S_{ji}} \frac{1}{SD(x)} = \sum_{x \in S_{ji}} \frac{1}{SD(x)} + \sum_{x \in S_{ij}} \frac{1}{SD(x)} = D_3(\#j, \#i)$$

c) Axiom α_3 is satisfied since:

According to *lemma 3.1* ,

$$S_{ij} \subseteq S_{ii} \cup S_{lj} \quad (\text{I})$$

Also, whenever $f(x) > 0$ for all x such that $x \in A \cup B$, it holds that:

$$\sum_{x \in A \cup B} f(x) = \sum_{x \in A} f(x) + \sum_{x \in B} f(x) - \sum_{x \in (A \cap B)} f(x) \leq \sum_{x \in A} f(x) + \sum_{x \in B} f(x) \quad (\text{II})$$

and

$$\sum_{x \in A} f(x) \leq \sum_{x \in A \cup B} f(x) \quad (\text{III})$$

Thus, since $\frac{1}{SD(x)} > 0$ for any $x \in C_i, i \geq 1$, (II) and (III) imply that

$$\sum_{x \in S_{ij}} \frac{1}{SD(x)} \leq \sum_{x \in (S_{ii} \cup S_{lj})} \frac{1}{SD(x)} \leq \sum_{x \in S_{ii}} \frac{1}{SD(x)} + \sum_{x \in S_{lj}} \frac{1}{SD(x)}$$

□

Theorem 3.5: *Function d_3 is a metric.*

PROOF: i) Since $D_3(\#i, \#j)$ has been shown by *theorem 3.3* to be a metric and the homographic transformation $\frac{aD}{1+aD}$ preserves the metric properties (due to *lemma 3.2*) d_3 is a metric for all $\#i, \#j$ that belong to IU.

ii) When $\#i, \#j$ belong to AU, d_3 is a metric since it equals to the identity function d_o , over the original classes $OC_{\#i}, OC_{\#j}$, which can be trivially shown [Gil87] to be a metric.

□

Theorem 3.6: *The semantic homogeneity of attributes is an equivalence relation*

PROOF: a) The semantic homogeneity is a reflexive relation (i.e. $sh(\#i, \#i)$) since:

$$OCL[\#i] \equiv OCL[\#i] \Rightarrow sh(\#i, \#j)$$

b) The semantic homogeneity is a symmetric relation (i.e. $sh(\#i, \#j) \iff sh(\#j, \#i)$), since:

$$sh(\#i, \#j) \Rightarrow OCL[\#i] = OCL[\#j] \Rightarrow OCL[\#j] = OCL[\#i] \Rightarrow sh(\#j, \#i)$$

c) The semantic homogeneity is a transitive relation (i.e. $sh(\#i, \#j)$ and $sh(\#j, \#l) \Rightarrow sh(\#i, \#l)$), since:

$$sh(\#i, \#j) \text{ and } sh(\#j, \#l) \iff (OCL[\#i] = OCL[\#j]) \text{ and } (OCL[\#j] = OCL[\#l]) \Rightarrow (OCL[\#i] = OCL[\#l]) \iff sh(\#i, \#l)$$

□

Theorem 3.7: *The attributes of an object #i, which are not semantically homogeneous to any attribute of an object #j belong to $A_{\#i}[c_k]$, for all c_k in $C[\#i, \#j]$*

PROOF:

$$(\forall x_1 : (x_1 \in INT[\#i]) \text{ and } (not(\exists x_2 : (x_2 \in INT[\#j]) \text{ and } sh(x_1, x_2)))) \Rightarrow$$

$$(not(\exists x_2 : (x_2 \in INT[\#j]) \text{ and } ((x_1, x_2) \in SH[\#i, \#j]))) \Rightarrow$$

$$(not(\exists x_2 c_k : (x_2 \in INT[\#j]) \text{ and } (c_k \in C[\#i, \#j]) \text{ and } ((x_1, x_2) \in c_k))) \Rightarrow$$

$$(\forall c_k : (c_k \in C[\#i, \#j]) \Rightarrow (x_1 \in A_{\#i}[c_k]))$$

□

Theorem 3.8: *Function D_4 is a metric.*

PROOF: a) D_4 satisfies axiom α_1 since:

i) when $INT[\#i] = \emptyset$ or $INT[\#j] = \emptyset$, $D_4(\#i, \#j) = \infty > 0$

ii) when $INT[\#i] \neq \emptyset$ and $INT[\#j] \neq \emptyset$,

Let us assume that at least one attribute, in each pair of attributes (x_1, x_2) in the optimal combination c_k between #i and #j has no attributes of its own. In this case $D_4(x_1, x_2, V) \geq 0$, as it was just proven, and by *lemmas 3.3* and *3.4* $D'(x_1, x_2, V) \geq 0$ and $d'(x_1, x_2, V) \geq 0$, respectively.

Furthermore by definition 3.34 $s(x) \geq 0$. Therefore,

$$\min_{c_k \in C[\#i, \#j]} \left\{ \sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 \right\} \geq 0$$

Thus, by *lemma 3.4* $d'(\#i, \#j, V) \geq 0$ and therefore $D_4(\#i, \#j, V) \geq 0$

b) D_4 satisfies axiom α_2 since:

i) When $INT[\#i] = \emptyset$ or $INT[\#j] = \emptyset$, we have: $D_4(\#i, \#j, V) = \infty = D_4(\#j, \#i, V)$

ii) When $INT[\#i] \neq \emptyset$ and $INT[\#j] \neq \emptyset$ we have:

$$\forall x_1, x_2 : ((x_1, x_2) \in SH[\#i, \#j]) \iff (x_1 \in INT[\#i]) \text{ and } (x_2 \in INT[\#j]) \text{ and } sh(x_1, x_2) \iff \\ (x_1 \in INT[\#i]) \text{ and } (x_2 \in INT[\#j]) \text{ and } sh(x_2, x_1) \iff (x_2, x_1) \in SH[\#j, \#i]$$

Therefore, a total and onto isomorphism f , can be defined between $SH[\#i, \#j]$ and $SH[\#j, \#i]$ such that: $f((x_1, x_2)) = (x_2, x_1)$

Consequently, each element c_k of $C[\#i, \#j]$ ($c_k \subseteq SH[\#i, \#j]$), will have a single image $f(c_k) = c'_k$ such that:

$$(c'_k \subseteq SH[\#j, \#i]) \text{ and } (\forall x_1, x_2 : ((x_1, x_2) \in c_k) \implies ((x_2, x_1) \in c'_k))$$

Since f is an isomorphism, it is inversible and f^{-1} is also a total and onto isomorphism.

Thus, each element c'_k of $C[\#j, \#i]$, will also have a single image $f^{-1}(c'_k) = c_k$ in $C[\#i, \#j]$.

Furthermore, for all c'_k , $A_{\#i}[c'_k] = A_{\#i}[c_k]$ because:

$$\forall x_1 : (x_1 \in A_{\#i}[c'_k]) \implies (\text{not } (\exists x_2, x_3 : ((x_2, x_3) \in c'_k \text{ and } (x_1 = x_3)) \implies \\ (\text{not } ((x_3, x_2) \in c_k) \text{ and } (x_1 = x_3))) \implies (x_1 \in A_{\#i}[c_k])$$

Similarly, for all c'_k , $A_{\#j}[c'_k] = A_{\#j}[c_k]$.

Let us also assume that at least one attribute of each pair of attributes (x_1, x_2) in the combinations c_k, c'_k has no attributes of its own. This is a valid assumption since eventually some of the $INT[x_1]$ or $INT[x_2]$ will become empty (even due to $V = IU \cup AU$). In this case, $D_4(x_1, x_2, V) = D_4(x_2, x_1, V)$ (as already proven) and therefore due to *lemma 3.4 d'* will be also symmetric: $d'(x_1, x_2, V) = d'(x_2, x_1, V)$

Thus, for all c_k, c'_k such that $f(c_k) = c'_k$:

$$\sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_2, V) = \sum_{(x_2, x_1) \in c'_k} s(x_2)s(x_1)d'(x_2, x_1, V)$$

and

$$\sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 = \sum_{x_3 \in A_{\#i}[c'_k]} s(x_3)^2 \text{ and } \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 = \sum_{x_4 \in A_{\#j}[c'_k]} s(x_4)^2$$

Therefore,

$$\sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_1, V) + \sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 = \\ \sum_{(x_2, x_1) \in c'_k} s(x_2)s(x_1)d'(x_2, x_1, V) + \sum_{x_3 \in A_{\#i}[c'_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c'_k]} s(x_4)^2$$

and thus

$$\min_{c_k \in C[\#i, \#j]} \left\{ \sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 \right\} =$$

$$\min_{c'_k \in C[\#j, \#i]} \left\{ \sum_{(x_2, x_1) \in c'_k} s(x_2)s(x_1)d'(x_2, x_1, V) + \sum_{x_3 \in A_{\#j}[c'_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#i}[c'_k]} s(x_4)^2 \right\}$$

or equivalently

$$D_4(\#i, \#j, V) = D_4(\#j, \#i, V) \quad (\text{II})$$

Thus, D_4 is also symmetric when both the involved objects have attributes.

c) D_4 satisfies axiom α_3 :

i) when two or three of the objects $\#i$, $\#j$ and $\#l$ have no attributes because:

$$D_4(\#i, \#j, V) \leq D_4(\#i, \#l, V) + D_4(\#l, \#j, V) \iff \infty \leq \infty + \infty \iff \infty \leq \infty$$

ii) when $\#i$ has no attributes the triangularity axiom is satisfied, because:

$$D_4(\#i, \#j, V) \leq D_4(\#i, \#l, V) + D_4(\#l, \#j, V) \iff \infty \leq \infty + D_4(\#l, \#j, V)$$

and $D_4(\#l, \#j, V) \geq 0$

iii) when $\#l$ has no attributes because:

$$D_4(\#i, \#j, V) \leq D_4(\#i, \#l, V) + D_4(\#l, \#j, V) \iff D_4(\#i, \#j, V) \leq \infty + \infty = \infty$$

and $D_4(\#i, \#j, V) \leq \infty$ by definition.

iv) when $\#j$ has no attributes, the triangularity axiom is satisfied, since:

$$D_4(\#i, \#j, V) \leq D_4(\#i, \#l, V) + D_4(\#l, \#j, V) \iff \infty \leq D_4(\#i, \#l, V) + \infty$$

and $D_4(\#i, \#l, V) \geq 0$

v) when all $\#i$, $\#j$ and $\#l$ have attributes since:

If C_{il}^{opt} and C_{lj}^{opt} are the optimal isomorphisms between the semantically homogeneous attributes of the objects $\#i$ $\#l$ and $\#l$, $\#j$, respectively, we can define C'_{ij} as an isomorphism derived from the transitivity of the semantic homogeneity relation (see *theorem 3.6*), as:

$$C'_{ij} = \left\{ (x_1, x_3) \mid (\exists x_2 : ((x_1, x_2) \in C_{il}^{opt}) \text{ and } ((x_2, x_3) \in C_{lj}^{opt})) \right\} \quad (\text{I})$$

Given the C'_{ij} , it is sufficient to prove that

$$\sum_{(x_1, x_2) \in C'_{ij}} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[C'_{ij}]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[C'_{ij}]} s(x_4)^2 \leq$$

$$\begin{aligned}
& \sum_{(x_5, x_6) \in C_{ij}^{opt}} s(x_5)s(x_6)d'(x_5, x_6, V) + \sum_{x_7 \in A_{\#i}[C_{ij}^{opt}]} s(x_7)^2 + \sum_{x_8 \in A_{\#i}[C_{ij}^{opt}]} s(x_8)^2 + \\
& \sum_{(x_9, x_{10}) \in C_{ij}^{opt}} s(x_9)s(x_{10})d'(x_9, x_{10}, V) + \sum_{x_{11} \in A_{\#i}[C_{ij}^{opt}]} s(x_{11})^2 + \sum_{x_{12} \in A_{\#j}[C_{ij}^{opt}]} s(x_{12})^2 \quad (II)
\end{aligned}$$

because even if C'_{ij} is not the optimal isomorphism C_{ij}^{opt} , between the objects $\#i$, $\#j$, C_{ij}^{opt} will result into a less total distance according to definition of function D_4 .

However, if:

$$C'_{il} = \left\{ (x_1, x_2) \mid ((x_1, x_2) \in C_{il}^{opt}) \text{ and } (\exists x_3 : ((x_1, x_3) \in C'_{ij})) \right\} \quad (III)$$

and

$$C'_{lj} = \left\{ (x_1, x_2) \mid ((x_1, x_2) \in C_{lj}^{opt}) \text{ and } (\exists x_3 : ((x_3, x_2) \in C'_{ij})) \right\} \quad (IV)$$

then,

$$\begin{aligned}
& \sum_{(x_1, x_2) \in C_{il}^{opt}} s(x_1)s(x_2)d'(x_1, x_2, V) = \\
& \sum_{(x_3, x_4) \in C'_{il}} s(x_3)s(x_4)d'(x_3, x_4, V) + \sum_{(x_5, x_6) \in (C_{il}^{opt} - C'_{il})} s(x_5)s(x_6)d'(x_5, x_6, V)
\end{aligned}$$

and

$$\begin{aligned}
& \sum_{(x_1, x_2) \in C_{lj}^{opt}} s(x_1)s(x_2)d'(x_1, x_2, V) = \\
& \sum_{(x_3, x_4) \in C'_{lj}} s(x_3)s(x_4)d'(x_3, x_4, V) + \sum_{(x_5, x_6) \in (C_{lj}^{opt} - C'_{lj})} s(x_5)s(x_6)d'(x_5, x_6, V)
\end{aligned}$$

Also, if we define:

$$A_{\#i}^{(1)}[C'_{ij}] = \left\{ x_1 \mid (\text{not } (\exists x_2 : (x_1, x_2) \in C_{il}^{opt})) \right\}$$

$$A_{\#i}^{(2)}[C'_{ij}] = \left\{ x_1 \mid (\exists x_2 : ((x_1, x_2) \in C_{il}^{opt})) \text{ and } (\text{not } (\exists x_3 : ((x_2, x_3) \in C_{ij}^{opt}))) \right\}$$

and

$$A_{\#j}^{(1)}[C'_{ij}] = \left\{ x_2 \mid (\text{not } (\exists x_1 : (x_1, x_2) \in C_{lj}^{opt})) \right\}$$

$$A_{\#j}^{(2)}[C'_{ij}] = \left\{ x_2 \mid (\exists x_1 : ((x_1, x_2) \in C_{lj}^{opt})) \text{ and } (\text{not } (\exists x_3 : ((x_3, x_1) \in C_{ij}^{opt}))) \right\}$$

then

$$A_{\#i}[C'_{ij}] = A_{\#i}^{(1)}[C'_{ij}] \cup A_{\#i}^{(2)}[C'_{ij}] \text{ and } A_{\#i}^{(1)}[C'_{ij}] = A_{\#i}[C_{il}^{opt}]$$

$$A_{\#j}[C'_{ij}] = A_{\#j}^{(1)}[C'_{ij}] \cup A_{\#j}^{(2)}[C'_{ij}] \text{ and } A_{\#j}^{(1)}[C'_{ij}] = A_{\#j}[C_{lj}^{opt}]$$

Thus, (II) becomes,

$$\begin{aligned} & \sum_{(x_1, x_2) \in C'_{ij}} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[C_{il}^{opt}]} s(x_3)^2 + \sum_{x_4 \in A_{\#i}^{(2)}[C'_{ij}]} s(x_4)^2 + \sum_{x_5 \in A_{\#j}[C_{lj}^{opt}]} s(x_5)^2 + \sum_{x_6 \in A_{\#j}^{(2)}[C'_{ij}]} s(x_6)^2 \leq \\ & \sum_{(x_7, x_8) \in C'_{ij}} s(x_7)s(x_8)d'(x_7, x_8, V) + \sum_{(x_9, x_{10}) \in (C_{il}^{opt} - C'_{il})} s(x_9)s(x_{10})d'(x_9, x_{10}, V) + \sum_{x_{11} \in A_{\#i}[C_{il}^{opt}]} s(x_{11})^2 + \sum_{x_{12} \in A_{\#i}[C_{il}^{opt}]} s(x_{12})^2 + \\ & \sum_{(x_{13}, x_{14}) \in C'_{il}} s(x_{13})s(x_{14})d'(x_{13}, x_{14}, V) + \sum_{(x_{15}, x_{16}) \in (C_{il}^{opt} - C'_{il})} s(x_{15})s(x_{16})d'(x_{15}, x_{16}, V) + \sum_{x_{17} \in A_{\#j}[C_{lj}^{opt}]} s(x_{17})^2 + \sum_{x_{18} \in A_{\#j}[C_{lj}^{opt}]} s(x_{18})^2 \quad (\text{V}) \end{aligned}$$

and equivalently,

$$\begin{aligned} & \sum_{(x_1, x_2) \in C'_{ij}} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_4 \in A_{\#i}^{(2)}[C'_{ij}]} s(x_4)^2 + \sum_{x_6 \in A_{\#j}^{(2)}[C'_{ij}]} s(x_6)^2 \leq \\ & \sum_{(x_7, x_8) \in C'_{il}} s(x_7)s(x_8)d'(x_7, x_8, V) + \sum_{(x_9, x_{10}) \in (C_{il}^{opt} - C'_{il})} s(x_9)s(x_{10})d'(x_9, x_{10}, V) + \sum_{x_{12} \in A_{\#i}[C_{il}^{opt}]} s(x_{12})^2 + \\ & \sum_{(x_{13}, x_{14}) \in C'_{il}} s(x_{13})s(x_{14})d'(x_{13}, x_{14}, V) + \sum_{(x_{15}, x_{16}) \in (C_{il}^{opt} - C'_{il})} s(x_{15})s(x_{16})d'(x_{15}, x_{16}, V) + \sum_{x_{18} \in A_{\#j}[C_{lj}^{opt}]} s(x_{18})^2 \quad (\text{VI}) \end{aligned}$$

However, according to the definition of the isomorphisms C'_{ij} , C'_{il} , C'_{lj} , we have that:

$$\forall x_1, x_3 : ((x_1, x_3) \in C'_{ij}) \Rightarrow (\exists x_2 : ((x_1, x_2) \in C'_{il}) \text{ and } ((x_2, x_3) \in C'_{lj})) \text{ and}$$

$$sh(x_1, x_3) \text{ and } sh(x_1, x_2) \text{ and } sh(x_2, x_3)$$

Then, according to the restrictions of the $s(x)$ in the definition of D_4 :

$$s(x_1) = s(x_2) = s(x_3) \quad (\text{VII})$$

Let us also assume that at least one attribute x of each pair of attributes in the combinations C'_{ij} , C'_{il} , C_{il}^{opt} , C'_{lj} , C_{lj}^{opt} has no attributes of its own. This is a valid assumption since eventually some of the $INT[x]$ will become empty (when $INT[x]$ will become empty or $V = IU \cup AU$). In this case, D_4 has been already shown to be triangular and therefore by *lemma 3.3* D' is also triangular, and furthermore by *lemma 3.4* (see below):

$$d'(x_1, x_3, V) \leq d'(x_1, x_2, V) + d'(x_2, x_3, V) \quad (\text{IIX})$$

Thus, from (VII), (IIX) we have that:

$$\begin{aligned} \sum_{(x_1, x_2) \in C'_{ij}} s(x_1)s(x_2)d'(x_1, x_2, V) &\leq \sum_{(x_1, x_3) \in C'_{ii}} s(x_1)s(x_3)d'(x_1, x_3, V) + \sum_{(x_2, x_3) \in C'_{ij}} s(x_2)s(x_3)d'(x_2, x_3, V) \iff \\ \sum_{(x_1, x_2) \in C'_{ij}} s(x_1)^2 d'(x_1, x_2, V) &\leq \sum_{(x_1, x_3) \in C'_{ii}} s(x_1)^2 d'(x_1, x_3, V) + \sum_{(x_2, x_3) \in C'_{ij}} s(x_1)^2 d'(x_2, x_3, V) \end{aligned}$$

Thus, for proving (VI) it is sufficient to prove that:

$$\begin{aligned} \sum_{x_4 \in A_{\#i}^{(2)}[C'_{ij}]} s(x_4)^2 + \sum_{x_6 \in A_{\#j}^{(2)}[C'_{ij}]} s(x_6)^2 &\leq \\ \sum_{(x_9, x_{10}) \in (C_{ij}^{opt} - C'_{ij})} s(x_9)s(x_{10})d'(x_9, x_{10}, V) + \sum_{x_{12} \in A_{\#l}[C_{ij}^{opt}]} s(x_{12})^2 + \\ \sum_{(x_{15}, x_{16}) \in (C_{ij}^{opt} - C'_{ij})} s(x_{15})s(x_{16})d'(x_{15}, x_{16}, V) + \sum_{x_{18} \in A_{\#l}[C_{ij}^{opt}]} s(x_{18})^2 &\quad (IX) \end{aligned}$$

However, due to definitions of $A_{\#i}^{(2)}[C'_{ij}]$, $A_{\#j}^{(2)}[C'_{ij}]$, $A_{\#l}[C_{ij}^{opt}]$ and $A_{\#l}[C_{ij}^{opt}]$, we have that:

$$\forall x_1 : (x_1 \in A_{\#i}^{(2)}[C'_{ij}]) \Rightarrow (\exists x_2 : ((x_1, x_2) \in C_{ij}^{opt}) \text{ and } (\text{not}(\exists x_3 : ((x_2, x_3) \in C_{ij}^{opt}))))$$

or, equivalently,

$$\forall x_1 : (x_1 \in A_{\#i}^{(2)}[C'_{ij}]) \Rightarrow (\exists x_2 : (x_2 \in A_{\#l}[C_{ij}^{opt}]) \text{ and } sh(x_1, x_2))$$

Then according to the restriction of the s(i) factors, it will also be that $s(x_1) = s(x_2)$ and therefore

$$\sum_{x_4 \in A_{\#i}^{(2)}[C'_{ij}]} s(x_4)^2 \leq \sum_{x_{18} \in A_{\#l}[C_{ij}^{opt}]} s(x_{18})^2 \quad (X)$$

Similarly,

$$\forall x_1 : (x_1 \in A_{\#j}^{(2)}[C'_{ij}]) \Rightarrow (\exists x_2 : ((x_2, x_1) \in C_{ij}^{opt}) \text{ and } (\text{not}(\exists x_3 : ((x_3, x_2) \in C_{ij}^{opt}))))$$

or equivalently,

$$\forall x_1 : (x_1 \in A_{\#j}^{(2)}[C'_{ij}]) \Rightarrow (\exists x_2 : (x_2 \in A_{\#l}[C_{ij}^{opt}]) \text{ and } sh(x_2, x_1))$$

Then according to the restriction of the s(i) factors, it will also be that

$$s(x_1) = s(x_2)$$

and therefore

$$\sum_{x_4 \in A_{\#i}^{(2)}[C'_{ij}]} s(x_4)^2 \leq \sum_{x_{18} \in A_{\#l}[C_{ij}^{opt}]} s(x_{18})^2 \quad (XI)$$

However, (X) and (XI) imply that (IX) and equivalently (II) are valid and therefore D_4 is triangular, even for objects, which have attributes. Thus, (IIX) is true in general and consequently D_4 will be also triangular in general.

□

Theorem 3.9: *Function d_4 is a metric.*

PROOF: This theorem is an implication of *theorem 3.8* and *lemma 3.2* .

□

Theorem 3.10: *All the pairs of attribute objects (x_1, x_2) of two objects $\#i, \#j$ that belong to the set $SH[\#i, \#j]$ and also have the same original class will also belong to the optimal isomorphism c_{opt} between the objects $\#i$ and $\#j$.*

PROOF: We will prove this theorem by assuming that

$$\exists x_1, x_2 : OC_{x_1} = OC_{x_2} \text{ and } (not((x_1, x_2) \in c_{opt})) \quad (\text{I})$$

where c_{opt} is the optimal isomorphism between the objects $\#i, \#j$.

By definition 3.34, we also have that:

$$\begin{aligned} \forall c_k : (c_k \in C[\#i, \#j]) \Rightarrow \\ \sum_{(x_1, x_2) \in c_k} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[c_k]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_k]} s(x_4)^2 \geq \\ \sum_{(x_1, x_2) \in c_{opt}} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#i}[c_{opt}]} s(x_3)^2 + \sum_{x_4 \in A_{\#j}[c_{opt}]} s(x_4)^2 \end{aligned} \quad (\text{II})$$

Suppose that, the optimal isomorphism c_{opt} maps an attribute x_n of object $\#i$ onto to attribute x_2 of object $\#j$, such that $x_1 \neq x_n$.

Then, if we define isomorphism c' as:

$$c' = c_{opt} - \left\{ (x_1, x_k), (x_n, x_2) \right\} \cup \left\{ (x_1, x_2), (x_n, x_k) \right\}$$

we can prove that:

$$s(x_1)s(x_2)d'(x_1, x_2, V) + s(x_n)s(x_k)d'(x_n, x_k, V) < s(x_1)s(x_k)d'(x_1, x_k, V) + s(x_n)s(x_2)d'(x_n, x_2, V)$$

or, equivalently, that:

$$d'(x_1, x_2, V) + d'(x_n, x_k, V) < d'(x_1, x_k, V) + d'(x_n, x_2, V) \quad (\text{III})$$

since $s(x_1) = s(x_2) = s(x_n) = s(x_k)$.

This equality holds because the attributes x_1, x_n and x_2, x_k are semantically homogeneous or otherwise they couldn't be mapped on to each other.

Inequality (III) will be true if:

$$D'(x_1, x_2, V) + D'(x_n, x_k, V) < D'(x_1, x_k, V) + D'(x_n, x_2, V) \quad (\text{IV})$$

since d' is defined as a homographic transformation of D' , which is known to be a monotonically increasing function.

This will be proven for the most general case where

$$D'(x_i, x_j, V) = D(x_i, x_j, V) = (\underline{PD}_{ac} \underline{W}_{ac} \underline{PD}_{ac}^T)^{1/2}$$

but similarly it can be proven for the other cases in the definition of D' .

In this case, we have

$$\begin{aligned} d'(x_1, x_2, V) + d'(x_n, x_k, V) < d'(x_1, x_k, V) + d'(x_n, x_2, V) &\iff \\ d_1(x_1, x_2)^2 + d_2(x_1, x_2)^2 + 36d_3(x_1, x_2)^2 + d_4(x_1, x_2, V)^2 + d(o(x_1).TO, o(x_2).TO, V)^2 + \\ 2d_2(x_1, x_2)d_4(x_1, x_2, V) + 12d_2(x_1, x_2)d_3(x_1, x_2) + 12d_3(x_1, x_2)d_4(x_1, x_2, V) + \\ d_1(x_n, x_k)^2 + d_2(x_n, x_k)^2 + 36d_3(x_n, x_k)^2 + d_4(x_n, x_k, V)^2 + d(o(x_n).TO, o(x_k).TO, V)^2 + \\ 2d_2(x_n, x_k)d_4(x_n, x_k, V) + 12d_2(x_n, x_k)d_3(x_n, x_k) + 12d_3(x_n, x_k)d_4(x_n, x_k, V) < \\ d_1(x_1, x_k)^2 + d_2(x_1, x_k)^2 + 36d_3(x_1, x_k)^2 + d_4(x_1, x_k, V)^2 + d(o(x_1).TO, o(x_k).TO, V)^2 + \\ 2d_2(x_1, x_k)d_4(x_1, x_k, V) + 12d_2(x_1, x_k)d_3(x_1, x_k) + 12d_3(x_1, x_k)d_4(x_1, x_k, V) + \\ d_1(x_n, x_2)^2 + d_2(x_n, x_2)^2 + 36d_3(x_n, x_2)^2 + d_4(x_n, x_2, V)^2 + d(o(x_n).TO, o(x_2).TO, V)^2 + \\ 2d_2(x_n, x_2)d_4(x_n, x_2, V) + 12d_2(x_n, x_2)d_3(x_n, x_2) + 12d_3(x_n, x_2)d_4(x_n, x_2, V) \end{aligned} \quad (\text{V})$$

However, since x_1 and x_2 have the same original class, we also have that,

$OC_{x_n} \neq OC_{x_2}$ (see axiom A.3.19) and, $OC_{x_1} \neq OC_{x_k}$ (see axiom A.3.18)

Therefore by definition 3.29, $d_3(x_1, x_2) = 0$ $d_3(x_1, x_k) = 1$ $d_3(x_n, x_2) = 1$

Then, (V) is equivalent to:

$$\begin{aligned} d_2(x_1, x_2)^2 + d_4(x_1, x_2, V)^2 + d(o(x_1).TO, o(x_2).TO, V)^2 + 2d_2(x_1, x_2)d_4(x_1, x_2, V) + \\ d_2(x_n, x_k)^2 + 36 + d_4(x_n, x_k, V)^2 + d(o(x_n).TO, o(x_k).TO, V)^2 + \\ 2d_2(x_n, x_k)d_4(x_n, x_k, V) + 12d_2(x_n, x_k)d_3(x_n, x_k) + 12d_3(x_n, x_k)d_4(x_n, x_k, V) < \end{aligned}$$

$$\begin{aligned}
& d_2(x_1, x_k)^2 + 36 + d_4(x_1, x_k, V)^2 + d(o(x_1).TO, o(x_k).TO, V)^2 + \\
& 2d_2(x_1, x_k)d_4(x_1, x_k, V) + 12d_2(x_1, x_k)d_3(x_1, x_k) + 12d_3(x_1, x_k)d_4(x_1, x_k, V) + \\
& d_2(x_n, x_2)^2 + 36 + d_4(x_n, x_2, V)^2 + d(o(x_n).TO, o(x_2).TO, V)^2 + \\
& 2d_2(x_n, x_2)d_4(x_n, x_2, V) + 12d_3(x_n, x_2)d_2(x_n, x_2) + 12d_3(x_n, x_2)d_4(x_n, x_2, V)
\end{aligned} \tag{VI}$$

which is true since:

$$\begin{aligned}
& d_2(x_1, x_2)^2 + d_4(x_1, x_2, V)^2 + d(o(x_1).TO, o(x_2).TO, V)^2 + 2d_2(x_1, x_2)d_4(x_1, x_2, V) + \\
& d_2(x_n, x_k)^2 + 36 + d_4(x_n, x_k, V)^2 + d(o(x_n).TO, o(x_k).TO, V)^2 + \\
& 2d_2(x_n, x_k)d_4(x_n, x_k, V) + 12d_2(x_n, x_k)d_3(x_n, x_k) + 12d_3(x_n, x_k)d_4(x_n, x_k, V) < 72
\end{aligned}$$

Since (VI) is true, (III) is true as well and therefore

$$\begin{aligned}
& \sum_{(x_1, x_2) \in c'} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#}[c']} s(x_3)^2 + \sum_{x_4 \in A_{\#}[c']} s(x_4)^2 < \\
& \sum_{(x_1, x_2) \in c_{opt}} s(x_1)s(x_2)d'(x_1, x_2, V) + \sum_{x_3 \in A_{\#}[c_{opt}]} s(x_3)^2 + \sum_{x_4 \in A_{\#}[c_{opt}]} s(x_4)^2
\end{aligned} \tag{VII}$$

which contradicts the condition (II).

Thus, assumption (I) cannot be true and therefore we have that: x_1 and x_2 belong to the optimal isomorphism c_{opt} .

□

Theorem 3.11: *Function D is a metric.*

PROOF: We will prove axioms α_1 , α_2 , α_3 and α_4 for the case where

$$W_{ec} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 \end{bmatrix}$$

Similar proofs can be constructed for the other matrices of the quadric form (i.e. W_{et} , W_{ac} and W_{at}).

In the case of W_{ec} we have, $PD_{ec} = [d_1(x_1, x_2) d_2(x_1, x_2) d_3(x_1, x_2) d_4(x_1, x_2, V)]$

Thus,

$$D(x_1, x_2, V) =$$

$$(d_1(x_1, x_2)^2 + d_2(x_1, x_2)^2 + d_3(x_1, x_2)^2 + d_4(x_1, x_2, V)^2 + 2d_2(x_1, x_2)d_3(x_1, x_2) + 2d_2(x_1, x_2)d_4(x_1, x_2, V) + 2d_3(x_1, x_2)d_4(x_1, x_2, V))^{1/2} =$$

$$(d_1(x_1, x_2)^2 + (d_2(x_1, x_2) + d_3(x_1, x_2) + d_4(x_1, x_2, V))^2)^{1/2}$$

a) Axiom α_1 is satisfied since:

$$\forall x_1, x_2 : (d_1(x_1, x_2) \geq 0) \text{ and } (d_2(x_1, x_2) \geq 0) \text{ and } (d_3(x_1, x_2) \geq 0) \text{ and } (d_4(x_1, x_2, V) \geq 0) \Rightarrow \\ (d_1(x_1, x_2)^2 + (d_2(x_1, x_2) + d_3(x_1, x_2) + d_4(x_1, x_2, V))^2)^{1/2} \geq 0$$

b) Axiom α_2 is satisfied since:

$$\forall x_1, x_2 : (d_1(x_1, x_2) = d_1(x_2, x_1)) \text{ and } (d_2(x_1, x_2) = d_2(x_2, x_1)) \text{ and} \\ (d_3(x_1, x_2) = d_3(x_2, x_1)) \text{ and } (d_4(x_1, x_2, V) = d_4(x_2, x_1, V)) \Rightarrow \\ (d_1(x_1, x_2)^2 + (d_2(x_1, x_2) + d_3(x_1, x_2) + d_4(x_1, x_2, V))^2)^{1/2} = \\ (d_1(x_2, x_1)^2 + (d_2(x_2, x_1) + d_3(x_2, x_1) + d_4(x_2, x_1, V))^2)^{1/2} \Rightarrow D(x_1, x_2, V) = D(x_2, x_1, V)$$

c) Axiom α_3 is satisfied since:

If we define the vector $V_{x_1 x_2}$ as:

$$V_{x_1 x_2} = [d_1(x_1, x_2) \quad d_2(x_1, x_2) + d_3(x_1, x_2) + d_4(x_1, x_2, V)]$$

Then, $D(x_1, x_2, V) = ||V_{x_1 x_2}||_2$ where $||*||_2$ is the Euclidean Norm[Gil87].

It is known[Gil87] that $||*||_2$ obeys the triangle inequality:

$$||V_{x_1 x_2} + V_{x_2 x_3}||_2 \leq ||V_{x_1 x_2}||_2 + ||V_{x_2 x_3}||_2 \quad (\text{I})$$

Thus, if $d_5(x_i, x_j, V) = d_2(x_i, x_j) + d_3(x_i, x_j) + d_4(x_i, x_j, V)$

due to relation (I) we have,

$$((d_1(x_1, x_2) + d_1(x_2, x_3))^2 + (d_5(x_1, x_2, V) + d_5(x_2, x_3, V))^2)^{1/2} \leq \\ (d_1(x_1, x_2)^2 + d_5(x_1, x_2, V)^2)^{1/2} + (d_1(x_2, x_3)^2 + d_5(x_2, x_3, V)^2)^{1/2} \iff \\ (d_1(x_1, x_2)^2 + d_1(x_2, x_3)^2 + d_5(x_1, x_2, V)^2 + d_5(x_2, x_3, V)^2 + 2d_1(x_1, x_2)d_1(x_2, x_3) + 2d_5(x_1, x_2, V)d_5(x_2, x_3, V))^{1/2} \leq \\ (d_1(x_1, x_2)^2 + d_5(x_1, x_2, V)^2)^{1/2} + (d_1(x_2, x_3)^2 + d_5(x_2, x_3, V)^2)^{1/2} \quad (\text{II})$$

However, since d_1 and d_5 are metrics themselves, we also have that:

$$d_1(x_1, x_3) \leq d_1(x_1, x_2) + d_1(x_2, x_3) \iff \\ d_1(x_1, x_3)^2 \leq d_1(x_1, x_2)^2 + d_1(x_2, x_3)^2 + 2d_1(x_1, x_2)d_1(x_2, x_3) \quad (\text{III})$$

and similarly,

$$d_5(x_1, x_3, V)^2 \leq d_5(x_1, x_2, V)^2 + d_5(x_2, x_3, V)^2 + 2d_5(x_1, x_2, V)d_5(x_2, x_3, V) \quad (\text{IV})$$

By taking the square roots of the sum of (III) and (IV) we have that,

$$(d_1(x_1, x_3)^2 + d_5(x_1, x_3, V)^2)^{1/2} \leq$$

$$(d_1(x_1, x_2)^2 + d_1(x_2, x_3)^2 + d_5(x_1, x_2, V)^2 + d_5(x_2, x_3, V)^2 + 2d_1(x_1, x_2)d_1(x_2, x_3) + 2d_5(x_1, x_2, V)d_5(x_2, x_3, V))^{1/2} \quad (\text{V})$$

Thus, from (II) and (V) we have that,

$$(d_1(x_1, x_3)^2 + d_5(x_1, x_3, V)^2)^{1/2} \leq (d_1(x_1, x_2)^2 + d_5(x_1, x_2, V)^2)^{1/2} + (d_1(x_2, x_3)^2 + d_5(x_2, x_3, V)^2)^{1/2} \iff$$

$$D(x_1, x_3, V) \leq D(x_1, x_2, V) + D(x_2, x_3, V)$$

d) Axiom α_4 is satisfied, since:

$$D(\#i, \#j) = 0 \iff$$

$$(d_1(\#i, \#j)^2 + d_2(\#i, \#j)^2 + d_3(\#i, \#j)^2 + d_4(\#i, \#j, V)^2 + 2d_2(\#i, \#j)d_3(\#i, \#j) + 2d_2(\#i, \#j)d_4(\#i, \#j, V) + 2d_3(\#i, \#j)d_4(\#i, \#j, V))^{1/2} = 0 \iff$$

$$(d_1(\#i, \#j) = 0 \text{ and } d_2(\#i, \#j) = 0 \text{ and } d_3(\#i, \#j) = 0 \text{ and } d_4(\#i, \#j, V) = 0 \text{ and } \iff \#i = \#j$$

□

Lemma 3.3: *The aggregate function D' is a metric.*

PROOF: It can be obtained exactly as the proof of *theorem 3.11* replacing D by D' .

□

Lemma 3.4: *Function d' is a metric .*

PROOF: This lemma is a consequence of *lemma 3.3*, stating that D' is a metric and *lemma 3.2*, stating that the homographic transformation of a metric is a metric itself.

□

Chapter 4

Saliency Functions

4.1 The Problem of Saliency

The distinction between salient and non-salient features is critical for successful analogical reasoning. This is because only analogies between salient features may lead to consistent and pragmatically useful (i.e. resolving problems) conjectures of knowledge from a source to a target analog. However, salient features have a less important role in analogical retrieval. During that stage of analogical reasoning non-salient features should be also taken into account, since they may constitute a large part in the description of the target analog [Gen88b, Nov88, Ross88, Seif88].

Due to these considerations, most of the computational models of analogical reasoning incorporate criteria and mechanisms for distinguishing between features of different saliency.

As summarized by table 2.3 in chapter 2, four different sources of information have been used for computing saliency, namely:

- feature properties that can be derived from the syntax of their representation (e.g. SME [FFG90]);

- direct estimates provided by users(e.g. ARCS[THING90],MACKBETH[Win80]);
- causal knowledge distinguishing between features important to the purpose of analogical reasoning(usually expressed by relations between features and attainment of goals or by using salient features as indices to descriptions of analogs in memory[Kol84]) and features unimportant (e.g. CARL[Bur86], MEDIATOR[KSS85], NLANG[Grein88b], PRODIGY[VC91], CADET[SN91], Protos[PBH90]); and,
- assessments about the results of reasoning sessions provided by some external observer(e.g. CBL4[Aha91],CBR+EBL[CPS91]).

A basic criticism to computational solutions based on the latter three sources of information concerns that they are either intensive with respect to the acquisition of causal knowledge, or they depend on user judgements. Causal knowledge may prove impractical in dealing with analogical reasoning outside particular problem solving contexts or across multiple domains, while direct user estimates and assessments about results of reasoning are always sensitive to subjective biases. Also, the specific distinctions attempted by proposed general models of analogical reasoning using syntactic aspects of the representation of features are rather weak, as discussed in chapter 2.

The solution for estimating salience, which we develop in this thesis is motivated by the requirement to use only knowledge which is inherent in the conceptual descriptions of analogs, without relying on any sort of a-priori special knowledge about what is salient or not or on information supplied directly by the user. The whole approach is based on the concept of *feature dominance*, which is defined as a compound property derived from three other primitive properties of features. These are *charactericity*, *abstractness* and *determinance*. All these primitive properties are defined by logical conditions on the representation of features in conceptual models.

Salience is then introduced as belief to the truth value of the dominance of a feature and thus it provides a graded alternative to the logical strictness of this concept.

4.2 Relevant Modeling Concepts

As already stated, the primitive properties, which determine the dominance of a feature are defined through certain conditions on representations of features in a conceptual model.

These conditions involve five general modeling concepts, which are meaningful for representation languages and data models supporting the abstractions of classification, generalization and attribution. These concepts are:

- 1) the class which first introduces an attribute in a conceptual schema here called as *original domain class* of an attribute;
- 2) the set of classes which an attribute applies to, here called as *scope* of an attribute;
- 3) the set of classes which either introduce or refine an attribute, here called as the *refining classes*;
- 4) the set of distinct classes which are used as class-ranges for an attribute in a conceptual schema, here called as *possible attribute ranges*; and,
- 5) the set of instances of a class which are also instances of at least one of its subclasses, here called as *shared extension* of a class.

Formally, these concepts are defined as follows:

Definition 4.1: *The original domain class of an attribute i , ODC_i is the domain class j of its original class*

$$j = o(OC_i).FROM$$

Definition 4.2: *The scope of an attribute i , $S[i]$ is defined as:*

$$S[i] = \left\{ x \mid ODC_i \in o(x).Isa \right\}$$

Definition 4.3: *The set of the refining classes of an attribute i , $R[i]$, is defined as:*

$$R[i] = \left\{ c \mid (c \in S[i]) \text{ and } (\exists x_1 : (x_1 \in o(c).A) \text{ and } (n(i) = n(x_1))) \right\}$$

Definition 4.4: *The set of the possible attribute ranges of an attribute i , $AR[i]$, is defined as:*

$$AR[i] = \left\{ x \mid (\exists x_1, x_2 : (x_1 \in R[i]) \text{ and } (x_2 \in o(x_1).A) \text{ and } (n(x_2) = n(i)) \text{ and } (o(x_2).TO = x)) \right\}$$

Definition 4.5: The shared extension of a class c , $EXT_s[c]$, is defined as:

$$EXT_s[c] = \left\{ x \mid (c \in o(x).In) \text{ and } (\exists y : (c \in o(y).Isa) \text{ and } (y \in o(x).In)) \right\}$$

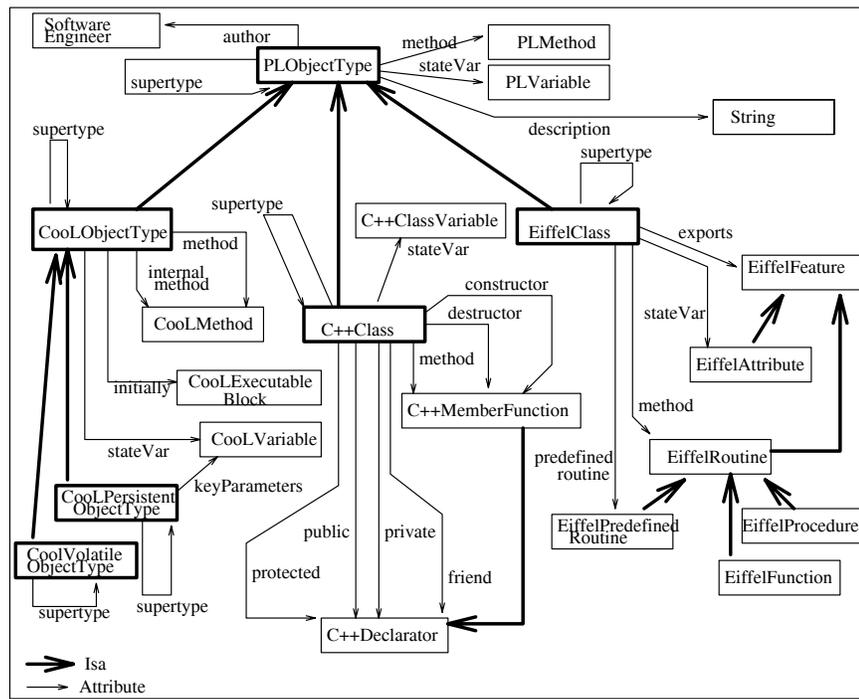


Figure 4.1: A Conceptual Schema for Object Types of OO Languages

Figure 4.1, presents a conceptual schema of object types in object-oriented programming languages (in the spirit of [Web92]). The original domain class of the attribute *supertype* of the class *CoolObjectType* is the class *PObjectType*. The scope of this attribute includes the classes: *PObjectType*, *CoolObjectType*, *CoolVolatileObjectType*, *CoolPersistentObjectType*, *C++Class* and *EiffelClass*. Also, the scope of the attribute *stateVar* of the class *PObjectType* comprises the same classes. However, the refining classes of these two attributes are different. While the refining classes of the attribute *supertype* coincide with the classes in its scope, the refining classes of the attribute *stateVar* include only the classes *PObjectType*, *CoolObjectType*, *C++Class* and *EiffelClass*. The possible ranges of the attribute *supertype* include the classes *PObjectType*,

CoolObjectType, *CoolVolatileObjectType*, *CoolPersistentObjectType*, *C++Class* and *EiffelClass*. Also, the possible ranges of the attribute *stateVar* include the classes *PLVariable*, *CoolVariable*, *C++ClassVariable* and *EiffelAttribute*.

4.3 The Dominance of Attributes

The concept of dominance is introduced as a compound property to express simultaneously:

- (i). the ability of an attribute to distinguish between classes of a conceptual schema and thus its importance to the construction of this schema;
- (ii). how essential an attribute may be for the behavior and the existence of the objects which are instantiated under the classes to which it applies; and,
- (iii). the ability of an attribute to determine the values of other attributes of the classes in its scope.

In this thesis, these three primitive properties of attributes are referred to as *charactericity*, *abstractness* and *determinance* of attributes, respectively.

4.3.1 The Charactericity of Attributes

The property of charactericity expresses whether or not an attribute can discriminate between the classes of its scope. The introduction of this property has been motivated by the formation of Isa hierarchies with strict inheritance in conceptual modeling. The classes of such hierarchies, are specialized when particular subsets of their extensions have one or more special properties. A special property may be an additional attribute or the restriction of the possible values of an existing attribute. Of course, the occurrence of such special properties does not necessarily lead to the specialization of the involved classes. In practice, only sets of properties of particular significance can dictate the specialization of a class, since it is necessary to avoid the cluttering of conceptual schemas with huge numbers of classes that should be added for each single new or refined attribute. The criteria enabling the decision on whether or not a class should be specialized, is not a matter of concern of this thesis.

Whenever a class must be specialized, this means introducing a new subclass of it. This subclass specifies a new attribute or refines an inherited one. Refinement assigns to an attribute a range class which is a subclass of the range class associated with it, in the superclass which is being specialized. New attributes characterize the subclass, with

respect to its specialized superclass. Refined attributes also differentiate the new subclass with respect to its superclass, but may not characterize it. This depends on whether or not other classes in the scopes of refined attributes, also associate these attributes with identical range classes. Attributes refined in many classes of an Isa hierarchy become characteristic of this hierarchy.

These considerations have motivated the following definition of the characteristicity of attributes:

Definition 4.6: *An attribute i is characteristic CH_i if and only if $|S[i]| = |AR[i]|$*

According to this definition, only attributes that have distinct range classes, for all the classes in their scope, are characteristic.

Attributes in figure 4.1 can be distinguished into characteristic and non characteristic, on the basis of definition 4.6 . For instance, the attribute *method*, which expresses the methods supported by an object type is characteristic, since it has distinct range classes in all the classes of its scope. Unlike it, the attribute *author* is not characteristic. The way it has been modeled, implies that nothing, worth representing in the conceptual schema, differentiates between the software engineers, who implement Eiffel, Cool and C++ object types.

4.3.2 The Abstractness of Attributes

The observation that attributes introduced in different classes in a conceptual schema, have different significance with regard to the existence and behavior of the objects which possess them motivates, the definition of the abstractness of attributes. In the object-oriented data modeling literature, classes are divided into *abstract* (also referred to as *virtual* [AH87] or *cluster heads* [Sci89]) and *concrete*, according to whether or not they have any instances of their own [Weg87, Joh88, Mey89, Sci89]. Abstract classes maintain only the really important attributes of their subclasses and suppress their detailed differences. Consider for example the difference in the significance of the attribute *hasSteeringSystem* of a vehicle and the attribute *luggageCarryingCapacity* of a car. The former is essential for driving any sort of vehicle (e.g. airplanes, trains, cars), while the latter neither applies to all vehicles (e.g. fighting airplanes) nor relates to the functionality of the special kind of vehicles which it applies to.

Attributes are distinguished into abstract and non-abstract according to whether or not the class that introduces them is abstract.

Formally, classes are distinguished into abstract and non-abstract(or concrete) by the following definition.

Definition 4.7: A class i is abstract AB_i , if and only if

$$EXT[i]=EXT_s[i]$$

On the basis of this definition, abstract attributes are defined as:

Definition 4.8: An attribute i is abstract (i.e. ABS_i) if and only if its original domain class u (i.e. $u = ODC_i$) is abstract (i.e. AB_u)

As an example, consider the taxonomy of resources(i.e. copies of books, cars and video tapes), which may be borrowed from some resource borrowing system(e.g. a library, a car rental agency and a video club), illustrated in figure 4.2.

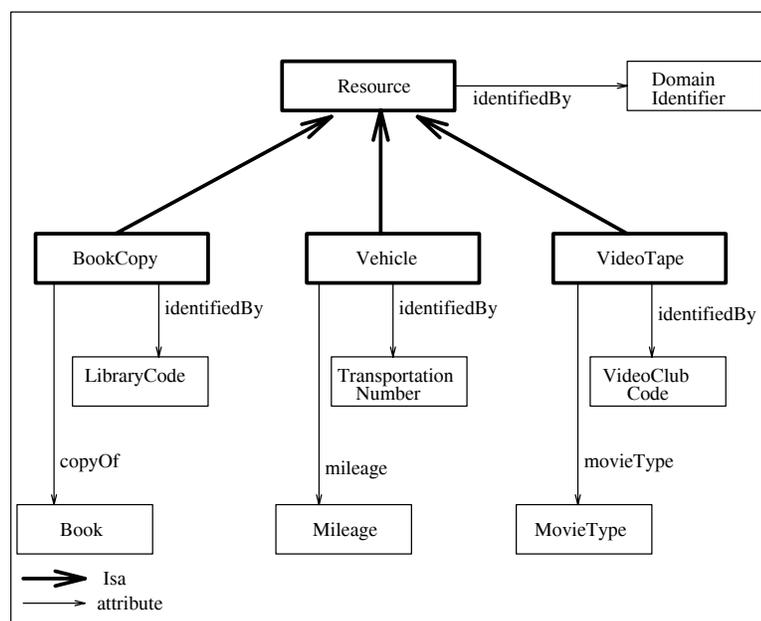


Figure 4.2: A Conceptual Schema of Borrowable Resources

The class *Resource* is an abstract one, since it can be reasonably assumed as being too general to have direct instances of its own. This class is specialized by three concrete classes, namely the classes *BookCopy*, *Car* and *VideoTape*, which express real resources

to be borrowed. The abstract class *Resource* and its concrete subclasses introduce attributes of different importance. For example, the attribute *identifiedBy* introduced by the class *Resource* is essential for any resource borrowing activity, regardless of the particular kind of resource, since it serves the necessity for a unique resource identification. Unlike it, the attributes *copyOf*, *mileage* and *movieType* are not essential for the basic function of resources as borrowable objects. Each of them represents only a piece of information used for locating a resource of a particular type. This difference is acknowledged in the schema by their introduction in concrete classes, as opposed to, the attribute *identifiedBy*, which is introduced in an abstract class.

4.3.3 The Determinance of Attributes

Determinance is introduced as a property expressing the ability of attributes to determine the values of other attributes, when they take certain values. For example, the model of a car could dictate the place where it has been produced, if the relevant manufacturer produces its different models, in different countries. Since determinance is crucial in inferencing, it has been realized in many different forms in analogical and case-based reasoning systems [Car83, Thag88, THNG90, SN91, CFR91]. Usually, such systems realize it from the pragmatic perspective of whether an attribute is related to a particular goal to be achieved. This pragmatic realization has been already referred to as the *pragmatic utility* of an attribute in the first chapter. A more general notion of determinance, similar to the one adopted here, has also been realized in the area of explanation-based learning, where predefined types of dependency relations are offered by languages for specifying explanations [Som88, Bar89, PBH90].

Here we concentrate on a special kind of dependencies, in order to avoid the cost of acquiring explicit knowledge about them. This acquisition would also increase the complexity of the underlying representation formalism, since it would require the existence of special constructs for representing such dependencies (e.g. rules in [CPS91], influences in [SN91]).

In particular, we concentrate on dependencies between attributes having the same domain as well as on a special case of them, *total equivalences*. Both these kinds of dependencies have the special property that conceptual schemas may imply their undefinability. Also, it must be pointed out that the *common domain condition* does not severely restrict the utility of the approach. This is because, as will become evident in the

following, the common domain condition requires that the involved attributes must be both applicable to at least one common class(see definition 4.9). However, this is not a limitation at least considering the problem-oriented representations of analogs in models of analogical reasoning. Such systems normally adopt description models, which aggregate all the attributes necessary for reasoning about an analog into a single class[Win80,FFG90,CPS91].

Suppose two attributes x and y are defined as associations of the following form:

$$x:D_x \rightarrow I_x \quad y:D_y \rightarrow I_y$$

where I_x and I_y denote the images(i.e. the sets of their actual values) of the attributes x and y . Attribute y can depend on attribute x (i.e. M_{xy}) only if they have identical domains, $D_x \equiv D_y$ (i.e. both of them apply to exactly the same set of objects). A dependency is then defined as a mapping $M:I_x \rightarrow I_y$. Total equivalences between x and y , hold in cases where M is a total and onto isomorphism.

The domain equality condition of dependencies may be checked in two ways. The first is to check whether the unions of the extensions of the classes in the scopes of two attributes i,j are identical(i.e. $\bigcup_{x \in S[i]} EXT[x] = \bigcup_{y \in S[j]} EXT[y]$). However, it may prove inadequate due to incomplete populations of objects in the relevant extensions or due to incorrect classifications of objects under any class in $S[i]$ or $S[j]$.

The alternative is to check whether two attributes have identical scopes(i.e. $S[i] = S[j]$). As it can be observed in figure 4.3, non common classes in the scopes of two attributes indicate that they have non identical domains(i.e. objects in $EXT[C1]-EXT[C2]$ and $EXT[C2]-EXT[C1]$), unless we assume that all the instances of class $C1$ must also be instances of class $C2$. In general, this cannot be precluded in representation languages, which allow multiple orthogonal instantiations of objects into classes not related by Isa relations. Nevertheless, it is rather unlikely. Hence, two classes not related by an Isa relation(directly or transitively) will not have identical extensions in general. Otherwise, there would be no need for maintaining both of them. It would be sufficient to have only one aggregating the attributes of both of them.

Accordingly, we will consider any dependency between attributes x and y as not definable(i.e. \bar{M}_{xy} or \bar{M}_{yx}) whenever:

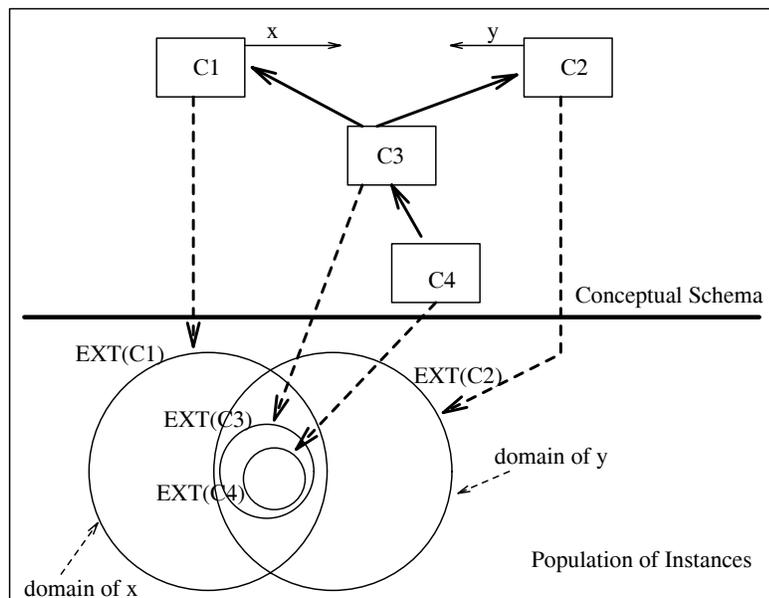


Figure 4.3: Non Common Classes of Attributes

$$(cnd1) : S[x] \neq S[y]$$

For the special kind of total equivalences, we can also identify a second definability condition. Given two attributes x and y , it will be impossible to define a total equivalence mapping between their images whenever their refining classes do not coincide:

$$(cnd2) : ((S[x] - R[x]) \cap R[y]) \cup ((S[y] - R[y]) \cap R[x]) \neq \emptyset$$

Figure 4.4 presents a case where this condition is not satisfied, since attribute x_1 is refined by class C_2 , but not attribute x_2 . If an equivalence mapping between these two attributes were definable their images with respect to C_1 and C_2 , should have equal numbers of elements (i.e. $|EXT(R_1)| = |EXT(R_3)|$ and $|EXT(R_2)| = |EXT(R_3)|$). Since, attribute x_1 is refined in C_2 , its image with respect to C_2 will have fewer elements than its image with respect to C_1 (i.e. $|EXT(R_2)| < |EXT(R_1)|$), which is a contradiction.

The definability conditions of interattribute dependencies are exemplified in figure 4.5. This figure presents a taxonomy of the different types of employees of a research institute. Given this taxonomy, the conditions $cnd1$ and $cnd2$ imply that no dependency mapping could be defined between the attributes *worksFor* of the class *ResearchStaff* and *qualification* of the class *ResearchInstituteEmployee* or vice versa. The first of these

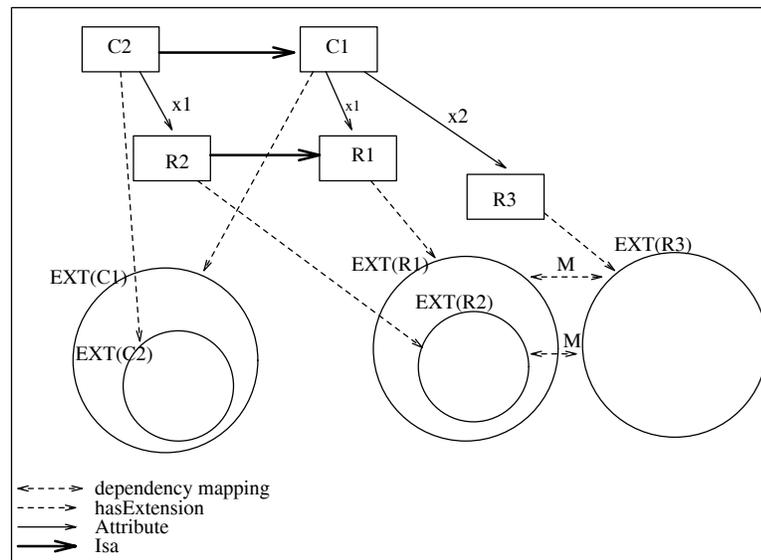


Figure 4.4: A Case of Non Common Refinement over the Conceptual Schema

attributes represents the relation between a researcher and the project (s)he is involved in, while the second represents the qualifications, that an employee should have for holding a certain position in the institute. Both conditions *cond1* and *cond2* are satisfied because these attributes do not have a common scope (i.e. the attribute *qualification* applies to all classes of employees while the attribute *worksFor* does not). and furthermore, they are refined in different classes.

By contrast a dependency between the attributes *worksFor* and *belongsTo* (it express the research group of a researcher) both introduced in the class *ResearchStaff* is possible, since these attributes have identical scopes and they are not refined separately. Failure to preclude the definability of a dependency mapping may be intuitively associated with a possible dependency between these two attributes. Projects are normally allocated to research groups and thus the projects a researcher is involved in will depend on his(her) group. Nevertheless, notice that failure to preclude the definability of a dependency between these attributes does not necessarily imply that such dependency exists.

On the basis of interattribute dependencies we define determinative attributes as:

Definition 4.9: An attribute with identifier i is determinative DET_i , if and only if $(\exists c, y : (c \in S[i]) \text{ and } (y \in INT[c]) \text{ and } M_{iy})$

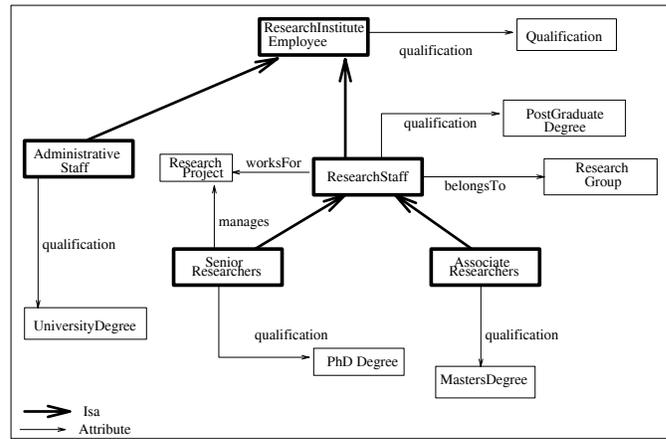


Figure 4.5: A Conceptual Schema of Employees of a Research Institute

According to this definition, it is sufficient to identify even one dependency between an attribute x and another attribute y of a class in its scope, for characterizing x as determinative.

4.3.4 The Compound Property of Dominance

The abstractness, charactericity and determinance of attributes determine their dominance, according to the following definition:

Definition 4.10: An attribute with identifier i is domain dominant DOM_i , if and only if $(ABS_i$ and $CH_i)$ or DET_i

The conjunction of abstractness and charactericity excludes cases satisfying only one of these properties, as extreme ones. In fact, attributes may be introduced in abstract classes only because of their broad applicability. For example, the attribute *description* of the class *PLObjectType* in figure 4.1, serves only as a textual annotation of object types with no essential implication for the existence or the behavior of their instances. Notice also, that attributes introduced in leaf classes of Isa hierarchies, although they satisfy the definition of charactericity, do not have a general classification significance for the entire taxonomies. This is because leaf classes represent only small subsets of larger sets of objects, which in turn are represented by classes higher up in Isa hierarchies.

the types in object-oriented programming languages, which are encapsulated. These are types which distinguish between two basic parts for their instances: a *private* or *encapsulated* part consisting of methods and state variables which cannot be accessed or called by other objects and a public part consisting of methods and/or state variables which are accessible by other objects (see [Mey89] for a detailed discussion on encapsulation in object-oriented programming). The existence of the class *PLEncapsulatedObjectType* violates the characteristicity condition with respect to the attribute *method* of the class *PLOBJECTType*. Nevertheless, this attribute still captures better than other attributes (e.g. the attribute *author* of the same class) the intuition behind the characteristic attributes. After all, it discriminates between most of the classes in its scope.

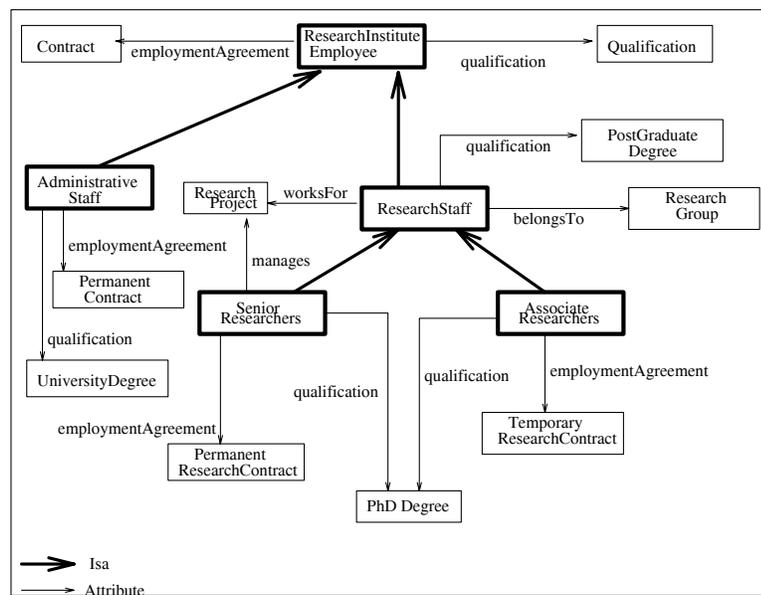


Figure 4.7: An Extended Conceptual Schema of Employees

Also, the condition of characteristicity can be violated due to schema incompleteness. An example of such a case is presented in figure 4.7, which extends the description of the research institute employees of figure 4.5, with an additional attribute expressing the contract of employment between an employee and the institute (i.e. the attribute *employmentAgreement* introduced by the class *ResearchInstituteEmployee*). This attribute fails to satisfy the characteristicity condition, since there exists in the schema a specialization of employees (i.e. the class *ResearchStaff*), which does not refine it. Notice that, the attribute

employmentAgreement could be refined by the class *ResearchStaff* so as to take values into a new class, the class *ResearchContract*, which would abstract the research work predicted by the employment contracts of all the employees belonging to the research staff, regardless of their temporary or permanent duration.

Violations of the Condition of Abstractness. The condition distinguishing between abstract and non abstract classes(i.e. $EXT_s[c] = EXT[c]$) may also be violated when objects are not classified under the most specific possible class in a conceptual schema.

Such classifications occur when the knowledge about one or more of the attributes of some object is not yet sufficient to instantiate it under a specialized class in some taxonomy. Thus, the object is directly instantiated under some more general class, which otherwise would be characterized as abstract. Over general classification may also occur when, although the available knowledge for an object could allow its instantiation under a specific class, the object is not instantiated under this class by mistake.

Accidental Satisfaction of the Conditions for Non Dependencies. Schema incompleteness may also result in an accidental satisfaction of the conditions implying the non definability of interattribute dependencies, defying intuition behind the relevant cases. Such an example is also presented in figure 4.7. In this figure, the non refinement of the attribute *employmentAgreement* in the class *ResearchStaff* causes the satisfaction of the non common refinement condition with respect to this attribute and the attribute *qualification*. Thus, a dependency mapping between these two attributes, would be precluded. However, it could be intuitively claimed that different kinds of employment agreements require certain types of qualifications from employees (e.g. a research contract presumes the possession of a postgraduate degree). Thus, a dependency mapping between these attributes is not unlikely.

All these examples suggest that exact reasoning about the truth value of the dominance of attributes may prove too restrictive due to imperfections of conceptual models. This problem can be addressed by inexact reasoning about the discussed properties of attributes(i.e. the logical inference of their truth values is replaced by an uncertain inference about them). This solution is articulated around the concept of attribute *salience*, which is discussed and defined in the following sections.

4.4 The Saliency of Attributes

The possible characteristics of conceptual models, that were outlined in the preceding section, question the reliability of the inferences about the logical properties of attributes. This is because they are checked using information which may not be fully reliable. To overcome this problem, the assessment of the dominance of attributes, could become an instance of numeric approximate reasoning[BK86]. Under this perspective, the inferences drawn about each of the three primitive properties of attributes as well as the property of dominance would be associated with some confidence measure about their truth values.

Such an uncertain approximation not only overcomes the problems that may arise due to imperfections of conceptual models but also allows even attributes with a low confidence about their dominance to influence similarity analysis and estimates, as suggested by relevant empirical studies[Nov88,SH88]. This is because it enables fine grain distinctions, rather than two-valued logical ones.

These considerations lead to defining the *saliency* of attributes as an interval measure of confidence that an attribute is dominant or not. The interval bounds result from combining functions, which in turn measure the extent to which, the logical conditions of charactericity, abstractness and determinance are satisfied.

In the literature, there have been proposed a considerable number of models for handling approximate reasoning, also referred to as *models of uncertainty*(see [BK86] for a theoretical and [NA90] for an application-oriented survey of these models). The more prominent among them include the *subjective probability theory*[DHN76,Quin83], the *possibility theory*[Zad86, Zad88], the *certainty factors model*[SB75,Hec86] and the *Dempster-Shafer theory of evidence*[Sha75].

The selection of an uncertainty model for handling the inference of saliency, is primarily motivated by two basic requirements for this problem, as it has been formulated so far.

The first requirement is imposed by the fact that the conditions implying the negation of a property do not always imply the property itself. This is the case with the determinance of attributes, which also affects the compound property of dominance. In such cases, a confidence measure c about the truth of a proposition does not always induce a confidence measure $(1-c)$ about the truth of its negation.

The second requirement concerns the interpretation of the used measures of confidence. Except from the possibility theory and the Dempster-Shafer theory, all the other referred models attempt to interpret their measures of uncertainty as *subjective probabilities* [SB75,DHN76,Quin83,Hec86]. Such an interpretation would inevitably require user supplied estimates of confidence, which have been a-priori precluded for pragmatic reasons in the present work. Also the combination of measures of confidence given probabilistic interpretations have been shown to violate certain axioms of probability theory[Quin83,Hec86].

These requirements, suggest the selection of the Dempster-Shafer theory, as the only model of handling uncertainty which satisfies them. According to it, the confidence about the properties of attributes is represented by two measures of *belief* and *plausibility*. Belief and plausibility of the dominance of an attribute can be estimated by combining partial beliefs to charactericity, abstractness and determinance, which in turn are measured by appropriate functions defined over a conceptual model. Before presenting the details of this approach, we introduce the basic terminology and the axiomatic foundation of the Dempster-Shafer theory of evidence.

4.4.1 Elements of the Dempster-Shafer Theory of Evidence

Belief in the context of the Dempster-Shafer theory of evidence is a number in the range $[0..1]$, reflecting the degree of support that a body of evidence provides for a proposition. Belief is formalized as a function $Bel: PowersetOf(\Theta) \rightarrow [0,..1]$, obeying the following axioms:

$$(A. 4.1) \quad Bel(P) = 0, \text{ if } P = \emptyset$$

$$(A. 4.2) \quad Bel(\Theta) = 1$$

$$(A. 4.3) \quad Bel\left(\bigcup_{i=1}^n P_i\right) \geq \sum_{I \subseteq \{1,2,\dots,n\}, I \neq \emptyset} (-1)^{|I|+1} Bel\left(\bigcap_{i \in I} P_i\right)$$

$$n = |PowersetOf(\Theta)| \text{ and } P_i \subseteq \Theta, (i=1,\dots,n)$$

In these axioms, Θ is a set of mutually exclusive propositions referred to as the *frame of discernment* and, P is a subset of Θ , which expresses the logical disjunction of its elements[Sha75].

Notice that, according to axiom A.4.3 the sum of beliefs to a proposition P and its negation \bar{P} may be less than 1 (see belief Bel_1 in the example below). This is because, due to the axioms A.4.1, A.4.2 and A.4.3, we have:

$$1 = Bel(\Theta) = Bel(P \cup \bar{P}) \geq Bel(P) + Bel(\bar{P}) - Bel(P \cap \bar{P}) = Bel(P) + Bel(\bar{P})$$

The belief committed to P , $Bel(P)$, results from accumulating the beliefs committed to its subsets. As such, it is called *total belief* and is distinguished from the so called [Sha75] *basic probability assignment* or *mass* to P . The basic probability assignment is the belief that somebody commits exactly to P and cannot be split to any subset of it. The basic probability assignment is defined as a function m , obeying the following axioms:

$$(A.4.4) \quad m: PowersetOf(\Theta) \rightarrow [0, \dots, 1]$$

$$(A.4.5) \quad m(\emptyset) = 0$$

$$(A.4.6) \quad \sum_{P \subseteq \Theta} m(P) = 1$$

All the subsets of Θ , which are assigned a non zero basic probability (i.e. $m(P) > 0$) are referred to as *focals* of the basic probability assignment m . Notice that axiom (A.4.6) allows basic probability assignments to have non disjoint focals or even focals related with a subset relation (see basic probability assignment m_2 in the example below).

Each basic probability assignment m is said [Sha75] to *induce* a unique belief function defined as:

$$Bel(A) = \sum_{B \subseteq A} m(B) \tag{4.1}$$

Bel equals m for singleton subsets of Θ but is greater than or equal to m for subsets of Θ that contain more than one elements.

When distinct bodies of evidence give rise to distinct basic probability assignments, these assignments can be combined into a single basic probability assignment, according to the following formula (known as the rule of the *orthogonal sum* [Sha75]):

$$m_1 \overset{-}{+} \overset{-}{m_2}(P) = \frac{\sum_{X \cap Y = P} m_1(X)m_2(Y)}{1 - k_0} \tag{4.2}$$

k_0 in formula (4.2) is a normalizing parameter which increases the belief to the non empty intersections of the focals of the basic probability assignments m_1 and m_2 and is

defined as[Sha75]:

$$k_0 = \sum_{Z \cap W = \emptyset, Z \subseteq \Theta, W \subseteq \Theta} m_1(Z)m_2(W) \quad (4.3)$$

The *rule of the orthogonal sum* can be applied as long as:

$$\sum_{A \cap B \neq \emptyset, A \subseteq \Theta, B \subseteq \Theta} m_1(A)m_2(B) < 1$$

This condition precludes the combination of conflicting basic probability assignments(i.e. assignments, one of which provides a degree of support of 1 to some proposition, while the other provides an equal degree to the negation of this proposition).

The total belief about a proposition P, Bel(P) does not reflect the extent to which somebody fails to doubt P. This is given by a third measure called *upper probability* (or *plausibility* [Sha75]), defined as:

$$P^*(P) = 1 - Bel(\bar{P}) = \sum_{B \subseteq \Theta} m(B) - \sum_{A \subseteq \bar{P}} m(A) = \sum_{B \cap P \neq \emptyset} m(B) \quad (4.4)$$

Since $\sum_{B \cap P \neq \emptyset} m(B) \geq \sum_{B \subseteq P} m(B)$ also $P^*(P) \geq Bel(P)$. Hence, for each proposition P, we have a range in which its belief falls, the range [Bel(P),...,P*(P)]. In essence, P*(P) reflects the total belief which has not been assigned to the negation of P.

Example. To clarify the previous definitions consider the following example.

Two agents express their evidence about the winner of a football tournament, that is to take place with the participation of three different teams (i.e teams t1, t2 and t3). A frame Θ discerning the potential winner of the tournament equals the set $\{T1, T2, T3\}$. T1, T2 and T3 express the mutually exclusive propositions that the winner of the tournament will be the team t1, t2 or t3, respectively.

The first agent judges the potential winner on the basis of the competence of the players and the general condition of each team, as a whole. On the basis of this evidence, he assigns basic probabilities to the various subsets of Θ , as indicated by column m_1 of table 4.1(the first column of this table includes all the possible subsets of Θ). Notice that, he expresses his ignorance about the winner by assigning a basic probability 0.3 to Θ , so as to satisfy axiom A.4.6 .

The second agent judges the potential winner by partially available historical data about the same tournament. He knows that the tournament was organized with the participation of exactly the same teams 10 times in the past. The winner in the last two times was team t1 and the winner in the time prior to them was either team t1 or team t3 but he does not remember well. Also, he does not remember the winner of the first 7 times. On the basis of this evidence, he assigns basic probabilities to the various subsets of Θ , as indicated by column m_2 .

Notice that, the basic probability assignment m_2 to $\{T1, T3\}$ (i.e. 0.1) is less than the assignment to the singleton T1(i.e 0.2). This is possible in Dempster-Shafer theory. The basic probability assignment to $\{T1, T3\}$ expresses the available direct belief to the occurrence of the subset as a whole, and it should not be confused as implicit belief to the occurrence of any of its elements separately. Also, the ignorance about the winner of the first seven organizations of the tournament is expressed by assigning the fraction of these times with respect to the total number of the tournament organizations to the whole frame Θ

Set	m_1	Bel_1	P^*_1	m_2	Bel_2	P^*_2	m_{12}	Bel_{12}	P^*_{12}
T1	0	0	.8	.2	.2	1	.218	.218	.833
T2	0	0	.8	0	0	.7	0	0	.583
T3	.2	.2	.5	0	0	.8	.167	.167	.417
T1 T2	.5	.5	.8	0	.2	1	.365	.583	.833
T1 T3	0	.2	1	.1	.3	1	.031	.417	1
T2 T3	0	.2	1	0	0	.8	0	.167	.782
T1 T2 T3	.3	1	1	.7	1	1	.219	1	1

Columns Bel_1 and Bel_2 present the total beliefs estimated by each of the available basic probability assignments, according to formula (4.1). Notice that, the total belief Bel_2 to the subset $\{T1, T3\}$ (i.e. 0.3) is greater than the basic probability assignment to the same

set(i.e. 0.1), as the sum of this assignment and the assignment to the singleton T1(i.e. 0.2). This measure expresses the total belief(i.e direct and implicit) to the occurrence of any of the elements in the set.

Notice also that, beliefs and basic probability assignments to the same singletons are equal, as it can be deduced from formula (4.6) and axiom A.4.5, since singletons have no non empty subsets.

Columns P^*_1 and P^*_2 present the plausibilities of the various subsets of Θ , estimated according to each of the available basic probability assignments, using formula (4.4). These estimates express the maximum possible amount of belief that can be associated with any subset of Θ , since it has not been associated with its complementary set. For example, team t1 has a plausibility of 1 according to the second agent, since he does not believe directly or implicitly that the tournament will be won by some of the t2 and t3 teams.

Column m_{12} gives the combined basic probability assignments of the two agents, according to formula (4.2). The combination of two basic probability assignments to a couple of different subsets of Θ assigns a basic probability to their intersection. For example, the combination of the assignments $m_1(\{T1, T2\})$ (i.e. 0.5) and $m_2(\{T1, T3\})$ (i.e. 0.1) assigns a basic probability to the set $\{T1\}$ (i.e. 0.05). This assignment is added to other combined basic probabilities assigned to the same set by the combination of m_1 with m_2 (e.g. $m_1(\{T1, T2\})$ and $m_2(\{T1\})$). In our example, m_1 and m_2 have two focals, whose intersection is empty(i.e focals T3 and T1 T3). The basic probability assignment to this empty intersection, resulting from the combination of m_1 with m_2 (i.e. 0.04) is used as the parameter k_o , in normalizing the other assignments according to formula (4.2). The combined basic probability assignments m_{12} are accumulated into beliefs and plausibilities, exactly as the primitive ones. The results of this accumulation are given by columns Bel_{12} and P^*_{12} .

4.4.2 The Saliency Measure

The definition and the combination of partial beliefs to characteristicity, abstractness and determinance is possible through the introduction of a common frame of discernment. Certain elements in such a frame correspond to these properties and other to their logical

combinations.

The Common Frame of Discernment. A frame of discernment is defined for each attribute i . This frame discerns the properties of characteristicity, abstractness and determinance as well as all their logical combinations (i.e. conjunctions and disjunctions between them and their negations). A rigorous notation would require denoting each such frame separately by Θ_i . However, since the following analysis concerns only one attribute, the relevant frame can be denoted simply by Θ , without loss of generality.

Θ consists of all the possible proposition vectors $[vc, va, vm_1, vm_2, \dots, vm_n]$, whose elements are variables indicating the truth (when a variable takes the value 1, by convention) or the falsity (when a variable takes the value 0, by convention) of characteristicity (variable vc) and abstractness (variable va) of the attribute i , as well as of the definability of a dependency mapping between i and the n other attributes, defined in the classes of its scope (variables vm_j). Each proposition vector represents the conjunction of its elements. For example, the vector $[1, 1, 0, \dots, 0]$ concerning an attribute i , expresses the proposition that i is characteristic and abstract, while no dependency mapping can be defined between itself and any other attributes defined in the classes of its scope. Any pair of these proposition vectors must differ in the value of at least one of their elements, or otherwise they would express the same compound proposition. Non identical proposition vectors reflect mutually exclusive compound propositions. For instance, the proposition vectors $[0, 1, 0, \dots, 0]$ and $[0, 0, 0, \dots, 0]$, concerning an attribute i , express the mutually exclusive propositions that: (1) i is not characteristic and determinative but it is an abstract attribute and (2) i is not characteristic, determinative or abstract attribute. A set of proposition vectors represents the disjunction of its elements.

Given Θ , the properties determining the dominance of i are defined as the following subsets of it:

$$CH_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vc = 1 \right\}$$

$$\overline{CH}_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vc = 0 \right\}$$

$$ABS_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid va = 1 \right\}$$

$$\overline{ABS}_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid va = 0 \right\}$$

$$DET_i \equiv \bigcup_{j=1}^n M_j \text{ where } M_j \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vm_j = 1 \right\}$$

$$\overline{DET}_i \equiv \bigcup_{j=1}^n \overline{M}_j \text{ where } \overline{M}_j \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vm_j = 0 \right\}$$

$$DOM_i \equiv (CH_i \cap ABS_i) \cup DET_i$$

$$\overline{DOM}_i \equiv (\overline{CH}_i \cup \overline{ABS}_i) \cap \overline{DET}_i$$

Notice that, if P is understood as a logical proposition then \overline{P} denotes its negation. On the other hand, if P is understood as a subset of Θ , then \overline{P} denotes its complement with respect to Θ , which is equivalent to its negation. The construction of this frame is based on the underlying assumption that all the elementary propositions reflected by the elements of each proposition vector are logically independent. This means that the truth value of any of the primitive properties underlying dominance cannot be deduced given the truth values of any of the other. In spite of its context-questionable validity (especially regarding interattribute dependency mappings), this assumption is usually made by exact and inexact reasoning systems while handling aspects of a problem, whose logical or statistical interactions are not a-priori known or practically identifiable.

The General Functional Form of Saliency. Assuming the existence of a frame of discernment Θ , for each attribute i , saliency is defined as:

Definition 4.11: *Saliency of attribute classes is a function*

$$SL: A_c \rightarrow [0, \dots, 1]$$

such that

$$\forall i: (i \in A_c) \rightarrow SL(i) \in [Bel(DOM_i), \dots, P^*(DOM_i)]$$

In definition 4.11, A_c is the set of all attribute classes, meta classes, meta meta classes and so on. $Bel(DOM_i)$ is the belief on the dominance of attribute class i and, $P^*(DOM_i)$ is the upper probability (or plausibility) of the dominance of attribute class i .

Saliency measures are used for weighting attributes while computing interobject distances. According to definition 3.34 in chapter 3, each attribute is weighted by the

maximum of the salience measures of the original classes of its own classes. Thus, the lack of salience measures for token attributes does not affect our ability to weight them in similarity comparisons.

The conditioning of the primitive properties of attributes, which determine their dominance, by the way they have been abstractly modeled in conceptual schemas, allows us to concentrate on the general characteristics of entire classes of attributes rather than on particular attribute instances of these classes. However, the properties themselves are properties of these instances. For instance, the abstractness of the class of attributes *steering-System* indicates the importance of particular steering systems of specific cars and other vehicles to their ability of moving.

The current operationalization of the similarity model estimates SL as the mean of the salience range:

$$SL(i) = \frac{Bel(DOM_i) + P^*(DOM_i)}{2} \quad (4.5)$$

As it will be evident in the following sections this function aggregates beliefs to charactericity and abstractness, which determine both the lower ($Bel(DOM_i)$) and the upper ($P^*(DOM_i)$) boundaries of the salience range and beliefs to non determinance, which determine only the upper boundary ($P^*(DOM_i)$).

In the following, we present four basic probability assignments, which allow the estimation of these boundaries.

4.4.3 Basic Probability Assignment to Charactericity

The examples of violation of the charactericity condition, presented in section 4.3.5, motivate the following definition of a measure of evidence about the charactericity of an attribute (as a plausible relaxation of definition 4.6):

Definition 4.12: *The evidence to the charactericity of an attribute i , m_i^{ch} is defined as*

$$m_i^{ch}(P) = \begin{cases} c_i & \text{if } P = CH_i \\ 1-c_i & \text{if } P = \overline{CH}_i \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq CH_i \text{ \& } P \neq \overline{CH}_i \end{cases}$$

where

$$c_i = \frac{|AR[i]|}{|S[i]|}$$

As the following theorem indicates, m_i^{ch} is a basic probability assignment to the characteristic of an attribute, according to the axiomatic definition of such assignments in the context of the Dempster-Shafer theory of evidence.

Theorem 4.1: *The evidence measure*

$$m_i^{ch}(P) = \begin{cases} c_i & \text{if } P = CH_i \\ 1-c_i & \text{if } P = \overline{CH}_i \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq CH_i \text{ \& } P \neq \overline{CH}_i \end{cases}$$

where $c_i = \frac{|AR[i]|}{|S[i]|}$

is a basic probability assignment.

Notice that, m_i^{ch} results in a basic probability assignment of 1 to the characteristic of attribute i , whenever definition 4.6 is satisfied ($|AR[i]| = |S[i]| \rightarrow c_i = 1$).

4.4.4 Basic Probability Assignment to Abstractness

Since an attribute has been defined as abstract by consequence of the abstractness of its original domain class, the evidence about its own abstractness can be estimated from the evidence about the abstractness of that class, according to the following definition.

Definition 4.13: *The evidence to the abstractness of an attribute i , m_i^a is defined as*

$$m_i^a(P) = \begin{cases} e_j & \text{if } P = ABS_i \\ 1-e_j & \text{if } P = \overline{ABS}_i \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq ABS_i \text{ \& } P \neq \overline{ABS}_i \end{cases}$$

where

$$j = ODC_i$$

$$e_j = \frac{|EXT_s[j]|}{g(EXT[j])} \text{ and,}$$

$$g(EXT[j]) = \begin{cases} |EXT[j]| & \text{if } EXT[j] \neq \emptyset \\ 1 & \text{if } EXT[j] = \emptyset \end{cases}$$

This measure of evidence to the abstractness of an attribute is a basic probability assignment in the context of the Dempster-Shafer theory of evidence, as indicated by the following theorem.

Theorem 4.2: *The evidence measure*

$$m_i^a(P) = \begin{cases} e_j & \text{if } P = ABS_i \\ 1-e_j & \text{if } P = \overline{ABS}_i \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq ABS_i \text{ \& } P \neq \overline{ABS}_i \end{cases}$$

where

$$j = ODC_i$$

$$e_j = \frac{|EXT_s[j]|}{g(EXT[j])} \text{ and,}$$

$$g(EXT[j]) = \begin{cases} |EXT[j]| & \text{if } EXT[j] \neq \emptyset \\ 1 & \text{if } EXT[j] = \emptyset \end{cases}$$

is a basic probability assignment.

The definition of measure m_i^a is motivated by a plausible relaxation of definition 4.7, which gives a belief of 1 whenever this definition is satisfied provided that $EXT[j] \neq \emptyset$ ($|EXT_s[j]| = |EXT[j]|$ & $EXT[j] \neq \emptyset \rightarrow e_j = 1$).

Furthermore, this measure is not as sensitive as definition 4.7 to cases where objects are instantiated under abstract classes in a conceptual schema, although they should be instantiated under non abstract ones, as discussed in section 4.3.5 .

4.4.5 Basic Probability Assignments to Non Determinance

Evidence about the non determinance of an attribute can be measured as an approximate assessment of the conditions *cmd1* and *cmd2* .

Basic Probability Assignment to Undefinability of Dependencies. In particular, evidence about the undefinability of a dependency mapping between two attributes *i* and *j*, due to their non common classes is measured according to the following definition.

Definition 4.14: *The evidence to the undefinability of a dependency mapping between an attribute *i* and an attribute *j*, due to their non common scope, m_{ij}^{ncs} , is defined as*

$$m_{ij}^{ncs}(P) = \begin{cases} d_{ij} & \text{if } P = \overline{M}_j \\ 1-d_{ij} & \text{if } P = \Theta \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq \overline{M}_j \text{ \& } P \neq \Theta \end{cases}$$

$$\text{where } d_{ij} = \frac{(|S[i]-S[j]|+|S[j]-S[i]|)}{(|S[i] \cup S[j]|)}$$

According to $m_{ij}^{ncs}(P)$, the more the non common classes in the scopes of two attributes the more evident the impossibility to define a dependency mapping between them.

The evidence measure m_{ij}^{ncs} is a basic probability assignment in the context of the Dempster-Shafer theory, as indicated by the following theorem.

Theorem 4.3: *The evidence measure*

$$m_{ij}^{ncs}(P) = \begin{cases} d_{ij} & \text{if } P = \bar{M}_j \\ 1-d_{ij} & \text{if } P = \Theta \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq \bar{M}_j \text{ \& } P \neq \Theta \end{cases}$$

$$\text{where } d_{ij} = \frac{|S[i]-S[j]| + |S[j]-S[i]|}{|S[i] \cup S[j]|}$$

is a basic probability assignment.

Basic Probability Assignment to Undefinability of Total Equivalences. The evidence about the undefinability of total equivalence mappings between two attributes i and j , due to their non common refinements over a conceptual schema is measured according to the following definition.

Definition 4.15: *The evidence to the undefinability of a total equivalence mapping between an attribute i and an attribute j , due to their non common refinements over a conceptual schema, m_{ij}^{ncr} , is defined as*

$$m_{ij}^{ncr}(P) = \begin{cases} r_{ij} & \text{if } P = \bar{M}_j \\ 1-r_{ij} & \text{if } P = \Theta \\ 0 & \text{if } (P \subseteq \Theta \text{ \& } P \neq \bar{M}_j \text{ \& } (P \neq \Theta)) \end{cases}$$

$$\text{where } r_{ij} = \frac{|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]|}{|R[i] \cup R[j]|}$$

According to m_{ij}^{ncr} , the more the non common refinements of two attributes over a schema, the more evident the impossibility of defining a total equivalence mapping between them.

The evidence measure m_{ij}^{ncr} is a basic probability assignment in the context of the Dempster-Shafer theory, as indicated by the following theorem.

Theorem 4.4: *The evidence measure*

$$m_{ij}^{ncr}(P) = \begin{cases} r_{ij} & \text{if } P = \bar{M}_j \\ 1-r_{ij} & \text{if } P = \Theta \\ 0 & \text{if } (P \subseteq \Theta \text{ \& } P \neq \bar{M}_j \text{ \& } P \neq \Theta) \end{cases}$$

$$\text{where } r_{ij} = \frac{|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]|}{|R[i] \cup R[j]|}$$

is a basic probability assignment.

Both $m_{ij}^{ncs}(\overline{M}_j)$ and $m_{ij}^{ncr}(\overline{M}_j)$ are 0, whenever the negations of conditions *cond1* and *cond2* are satisfied since:

$$S[i] = S[j] \rightarrow d_{ij} = 0 \text{ and,}$$

$$((S[i]-R[i]) \cap R[j]) \cup ((S[j]-R[j]) \cap R[i]) = \emptyset \rightarrow r_{ij} = 0$$

Thus, they completely disfavor the undefinability of dependency or total equivalence mappings between values of attributes when *cond1* and *cond2* are not satisfied, yet they do not provide evidence favoring such mappings.

4.4.6 Total Belief and Plausibility of Dominance

As already mentioned, the basic probability assignments m_i^{ch} , m_i^a , m_{ij}^{ncs} and m_{ij}^{ncr} provide evidence to subsets of the introduced frame of discernment, which correspond to the primitive properties of charactericity, abstractness, their negations as well as to non determinance of attributes.

The next step is to accumulate the partial beliefs provided by these assignments into a total belief and a plausibility measure about the dominance of an attribute *i* (DOM_i).

The Combinability of the Basic Probability Assignments. As the following theorem indicates, the basic probability assignments m_i^{ch} , m_i^a , m_{ij}^{ncs} and m_{ij}^{ncr} satisfy the combinability condition of the Dempster-Shafer theory and thus, they can be combined to provide total belief measures. This condition requires that any two or more basic probability assignments must have *cores* (i.e. the unions of their focals), whose intersection is not empty for being combinable.

Theorem 4.5: *The basic probability assignments m_i^{ch} , m_i^a , m_{ij}^{ncs} and m_{ij}^{ncr} can be combined into total beliefs in any possible order.*

Functional Forms of Total Belief and Plausibility . The total belief to the non determinance of an attribute *i*, $Bel(\overline{DET}_i)$ can be obtained by considering only the *n* basic probability assignments m_{ij}^{ncs} or by considering both the *n* assignments m_{ij}^{ncs} and the *n* assignments m_{ij}^{ncr} . In both cases *n* is the number of the other attributes defined in the classes of the scope of *i*. The total belief resulting in the first case, may be used when every kind of

interattribute dependencies is taken into account, while the total belief resulting in the second case, may be used when only total equivalences between attributes are taken into account:

Theorem 4.6: a) The combination of the basic probability assignments m_{ij}^{ncs} , $j=1,2,\dots,n$ provides total belief to the non determinance of an attribute i , \overline{DET}_i estimated by:

$$Bel_1(\overline{DET}_i) = \prod_{j=1}^n d_{ij}$$

b) The combination of the basic probability assignments m_{ij}^{ncs} and m_{ij}^{ncr} , $j=1,2,\dots,n$ provides total belief to the non determinance of an attribute j , \overline{DET}_i estimated by:

$$Bel_2(\overline{DET}_i) = \prod_{j=1}^n (1 - (1 - r_{ij})(1 - d_{ij}))$$

The total belief and the plausibility of the dominance of an attribute i , can be obtained by combining the basic probability assignments m_i^{ch} and m_i^a with the assignments m_{ij}^{ncs} . The resulting belief and plausibility measures may be used when every kind of interattribute dependency is taken into account:

Theorem 4.7: The combination of the basic probability assignments m_i^a, m_i^{ch} , and m^{ncs} results into the following combined beliefs:

$$Bel(DOM_i) = c_i e_u$$

$$Bel(\overline{DOM}_i) = (1 - c_i e_u) \prod_{j=1}^n d_{ij}$$

$$P^*(DOM_i) = 1 - (1 - c_i e_u) \prod_{j=1}^n d_{ij}$$

where $u = ODC_i$.

Also, the total belief and the plausibility of the dominance of an attribute i , can be obtained by combining the basic probability assignments m_i^{ch} and m_i^a with the assignments m_{ij}^{ncs} and m_{ij}^{ncr} . The resulting belief and plausibility measures may be used when only interattribute total equivalences are taken into account:

Theorem 4.8: The combination of the basic probability assignments m_i^a, m_i^{ch} with the basic probability assignments m_{ij}^{ncs} and m_{ij}^{ncr} results into the following combined beliefs:

$$Bel(DOM_i) = c_i e_u$$

$$Bel(\overline{DOM}_i) = (1 - c_i e_u) \prod_{j=1}^n (1 - (1 - r_{ij})(1 - d_{ij}))$$

$$P^*(DOM_i) = 1 - (1 - c_i e_u) \prod_{j=1}^n (1 - (1 - r_{ij})(1 - d_{ij}))$$

where $u = ODC_i$.

4.5 Incorporation of Further Belief Measures

The basic probability assignments that were defined in the preceding sections can be viewed as a principled way of obtaining evidence from a conceptual model about the truth value of the three primitive properties of attributes.

In essence, these assignments attempt to objectify an act of judgement over the evidence provided by a conceptual model, related to the conditions which either define the relevant properties or imply their negations. The whole approach is based on the assumption, that conceptual models are sources of evidence about the dominance of attributes, since they comprise a set of modeling decisions about them, which reflect their underlying semantics[SC93,SC94c].

However, this approach by no means precludes the incorporation of further belief measures about the discussed properties of attributes, provided by the user. The only prerequisite for such additional beliefs, would be their reliance on entirely distinct bodies of evidence from the ones already taken into account by the introduced assignments(i.e. the conceptual model), which is a prerequisite for the applicability of the rule of the orthogonal sum[Sha75].

In the following, we show how two particular classes of basic probability assignments to the determinance and/or the non determinance of an attribute could be combined with the existing assignments. We also show the effect of potential combinations to the boundaries of the salience range. The investigation into the combination of additional beliefs to determinance is motivated by the fact that the assignments m_{ij}^{ncs} and m_{ij}^{ncs} provide only weak evidence about the non definability of dependency mappings, and they are not concerned with causal factors that could imply such dependencies or definitely preclude them.

The whole analysis concerns the classes of *simple support* and *bayesian belief* functions[Sha75]. Simple support functions are basic probability assignments which

assign a belief c to a single subset P of a frame of discernment Θ and the rest of it (i.e. $1-c$) to the entire frame of discernment. Thus, they are adequate for expressing beliefs in cases where evidence in favor of a proposition does not disfavor its negation. The functions m_{ij}^{ncs} and m_{ij}^{ncr} , which assign belief to the non definability of interattribute dependencies are simple support functions.

On the other hand, bayesian functions are basic probability assignments which satisfy the *additivity axiom* of the classical probability theory (i.e. $s(P)+s(\bar{P})=1$). Notice that the additivity axiom in probability theory is a special case of the additivity axiom (i.e. axiom A.4.3) in the Dempster-Shafer theory [Sha75]. Bayesian functions assign simultaneously a belief c to some subset P of a frame Θ and the rest of it ($1-c$) to its complement (i.e. \bar{P}). The function m_i^a is a bayesian belief function since it assigns a belief equal to e_j to the abstractness of an attribute and the rest of it (i.e. $1-e_j$) to the negation of this proposition.

These two classes of basic probability assignments cover a broad range of types of uncertainty about some proposition, as pointed out in the literature [SB75, Sha75, Quin83, BK86].

Incorporation of a Simple Support Function to the Determinance of an Attribute. A simple support basic probability assignment to the determinance of an attribute i , can be combined with the basic probability assignments m_i^{ch} , m_i^a and, m_{ij}^{ncs} to provide updated total belief and plausibility measures of its dominance:

Theorem 4.9: *The combination of any simple support basic probability assignment to the determinance of an attribute i , defined as:*

$$s(P) = \begin{cases} b & \text{if } P = \bigcup_{j=1}^n M_j \\ 1-b & \text{if } P = \Theta \end{cases}$$

where $0 \leq b \leq 1$ can be combined with the basic probability assignments m_i^{ch} , m_i^a and, m_{ij}^{ncs} only if:

$$b \prod_{j=1}^n d_{ij} < 1$$

Their combination, when valid, results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = e_u c_i + (1 - e_u c_i) \frac{b(1 - \prod_{j=1}^n d_{ij})}{1 - b \prod_{j=1}^n d_{ij}}$$

$$P^*(DOM_i) = 1 - (1 - e_u c_i) \prod_{j=1}^n d_{ij} \frac{1 - b}{1 - b \prod_{j=1}^n d_{ij}}$$

where $u = ODC_i$.

The combination of such a support function will increase both the belief and the plausibility of the dominance of the attribute in hand:

Theorem 4.10: *The combination of any simple support basic probability assignment to the determinance of an attribute i combinable with the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} will result in belief and plausibility measures of DOM_i , $Bel'(DOM_i)$ and $P'^*(DOM_i)$ such that:*

$$Bel'(DOM_i) \geq Bel(DOM_i)$$

$$P'^*(DOM_i) \geq P^*(DOM_i)$$

where $Bel(DOM_i), P^*(DOM_i)$ are the belief and the plausibility of DOM_i estimated from the combination of only the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} .

The shifting of the entire belief range will increase the salience measure of attribute i , due to formula 4.5.

Incorporation of a Simple Support Function to Non Determinance of an Attribute.

A simple support basic probability assignment to the non determinance of an attribute i , can be combined with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} to provide updated total belief and plausibility measures of its dominance:

Theorem 4.11: *The combination of any simple support basic probability assignment to the non determinance of an attribute i , defined as:*

$$s(P) = \begin{cases} b & \text{if } P = \bigcap_{j=1}^n \bar{M}_j \\ 1-b & \text{if } P = \Theta \end{cases}$$

where $0 \leq b \leq 1$ with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = e_u c_i$$

$$P^*(DOM_i) = 1 - (1 - e_u c_i) \left(\prod_{j=1}^n d_{ij} + b \prod_{j=1}^n (1 - d_{ij}) \right)$$

where $u = ODC_i$.

The combination of such a support function will not affect the belief to the dominance of the involved attribute but it will reduce the plausibility of its dominance:

Theorem 4.12: *The combination of any simple support function about the non determinance of an attribute i with the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} will result in belief and plausibility measures of the DOM_i , $Bel'(DOM_i)$ and $P'^*(DOM_i)$ such that:*

$$Bel'(DOM_i) = Bel(DOM_i) \text{ and } P'^*(DOM_i) \leq P^*(DOM_i)$$

where $Bel(DOM_i), P^*(DOM_i)$ are the belief and the plausibility of DOM_i estimated from the combination of only the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} .

Thus, it may reduce the salience of the involved attribute due to formula (4.5).

Incorporation of a Bayesian Function to the Determinance of an Attribute. A bayesian basic probability assignment to the determinance and the non determinance of an attribute i , can be combined with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} to provide updated total belief and plausibility measures of its dominance:

Theorem 4.13: *A bayesian basic probability assignment to the determinance and the non determinance of an attribute i , defined as:*

$$s(P) = \begin{cases} b & \text{if } P = \bigcap_{j=1}^n \bar{M}_j \\ 1-b & \text{if } P = \bigcup_{j=1}^n M_j \end{cases}$$

where $0 \leq b \leq 1$ is combinable with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} only if:

$$(1-b) \prod_{j=1}^n d_{ij} < 1$$

If combinable, their combination results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = \frac{e_u c_i b + (1-b)(1 - \prod_{j=1}^n d_{ij})}{1 - (1-b) \prod_{j=1}^n d_{ij}}$$

$$P^*(DOM_i) = 1 - \frac{(1 - e_u c_i) b}{1 - (1-b) \prod_{j=1}^n d_{ij}}$$

where $u = ODC_i$.

According to the combinability condition of this theorem, bayesian basic probability assignments, which assign a belief of 1 to the determinance of an attribute, cannot be combined with the basic probability assignments m_{ij}^{ncs} whenever they also assign a belief of 1 to the non determinance of that attribute, since they completely contradict each other. Notice also that the direction of the effect(i.e. increment or decrement) to the boundaries of the salience range, caused by such an incorporation, cannot be a-priori determined as in the case of the simple support functions.

4.6 An Example of Salience Estimation

In this section, we give an example of salience estimates, which are computed from the conceptual schema of figure 4.8. This schema extends the schema of figure 4.5 with more types of employees of a research institute and more attributes of these types. The thick boxes in the figure reflect the classes of employees, and the numbers in their right lower corners are the assumed numbers of the instances of these classes.

Every employee has a social security number(i.e. the attribute *securityNumber*) and a legal document determining the type of employment (i.e. the attribute *employmentAgreement*). Also, (s)he needs some qualifications(i.e. the attribute *qualification*) for occupying a certain position in the institute. Special types of employees may have additional attributes. Given this conceptual schema, the salience boundaries of the attributes have been computed as presented in table 4.2, without using any user-supplied belief functions in addition to those defined in this chapter.

The column $Bel(DOM_i)$ gives the lower bound of the total belief on an attribute's dominance. Hence, it may be used as a pessimistic estimate of salience, since it has been estimated without any source of evidence favoring the determinance of the attribute in hand. According to this lower bound, the most salient attribute is the

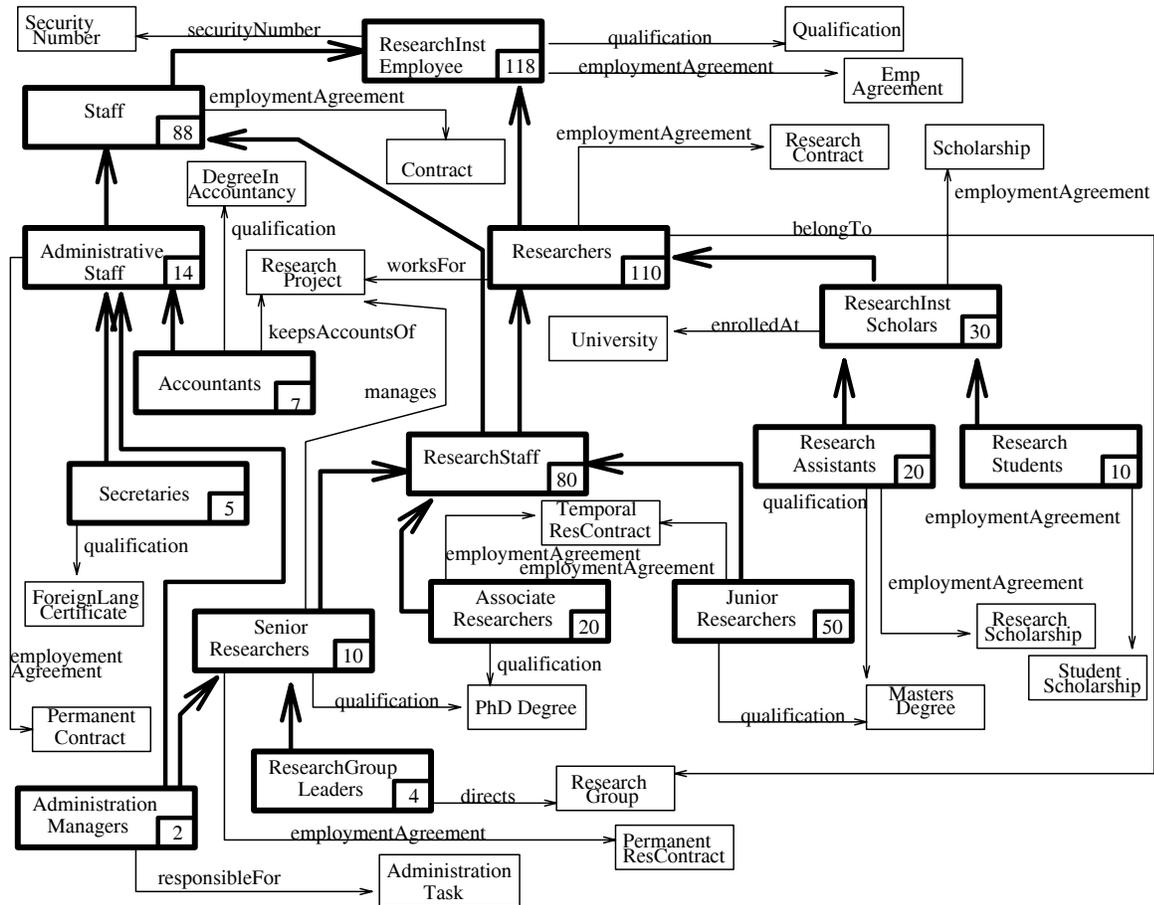


Figure 4.8: An Example for Estimating Saliency of Attributes

employmentAgreement. It can be intuitively claimed that this attribute gives rise to a taxonomy of employees, important for the institute, to the extent that the schema of figure 4.8 accurately represents, what is important. At the other end of the spectrum, the attribute *keepsAccountsOf* is not essential for the employment, since it applies only to a certain kind of employees.

The columns $P_1^*(DOM_i)$ and $P_2^*(DOM_i)$ of table 4.2 give the upper bounds of the beliefs on dominance, when all kinds of dependencies or only total equivalences are taken into account, respectively. These estimates could be used as optimistic measures of saliency, since they express the degree to which we fail to doubt about the dominance of attributes, according to the general interpretation of plausibility in [Sha75].

Table 4.2: Lower and Upper Boundaries of Salience							
Attribute	$Bel(CH_i)$	$Bel(ABS_i)$	$Bel_1(\overline{DET}_i)$	$Bel_2(\overline{DET}_i)$	$Bel(DOM_i)$	$P^*_1(DOM_i)$	$P^*_2(DOM_i)$
employmentAgreement	0.53	1	0	0.17	0.53	1	0.92
qualification	0.4	1	0	0.15	0.4	1.0	0.91
enrolledAt	0.33	1	0.25	0.47	0.33	0.83	0.68
manages	0.33	0.6	0.1	0.28	0.198	0.92	.77
worksFor	0.1	1	0	0	0.1	1.0	1.0
belongsTo	0.1	1	0	0	0.1	1.0	1.0
securityNumber	0.06	1	0	0	0.06	1.0	1.0
responsibleFor	1	0	0.44	0.64	0	0.56	0.36
directs	1	0	0.44	0.64	0	0.56	0.36
keepsAccountsOf	1	0	0.81	0.77	0	0.19	0.23

It is worth noticing that complete certainty about the dominance of the attributes *worksFor* and *belongsTo* (expressing the project and the groups which a researcher may belong to, respectively) cannot be precluded. This is because both these attributes are not refined separately over the schema and they do not have non common classes. Thus, the definability of a dependency mapping between them, cannot be a-priori precluded. In fact, it can be intuitively assumed that the participation of a researcher in some research group, determines the project in which (s)he is involved with.

4.7 Mathematical Appendix: Proofs of Theorems in Chapter 4

Theorem 4.1: *The evidence measure*

$$m_i^{ch}(P) = \begin{cases} c_i & \text{if } P = CH_i \\ 1-c_i & \text{if } P = \overline{CH_i} \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq CH_i \text{ \& } P \neq \overline{CH_i} \end{cases}$$

where $c_i = \frac{|AR[i]|}{|S[i]|}$, is a basic probability assignment.

PROOF: It is sufficient to prove that axioms (A.4.4), (A.4.5) and (A.4.6) are satisfied by the function m_i^{ch} .

a) Axiom (A.4.4) is satisfied since:

When $P \neq CH_i$ or $P \neq \overline{CH_i}$, $m_i^{ch}(P) = 0$ by definition.

When $P = CH_i$, we have that $m_i^{ch} = \frac{|AR[i]|}{|S[i]|}$

However, according to the definition of the set $R[i]$ (i.e. definition 4.3), it holds that:

$$\forall x : x \in R[i] \rightarrow x \in S[i]$$

Thus, $R[i] \subseteq S[i]$ and $|R[i]| \leq |S[i]|$ (I).

Also, according to the definition of the set $AR[i]$ (i.e. definition 4.4) we have that:

$$\forall x : (x \in AR[i]) \rightarrow (\exists y, z : (y \in R[i]) \text{ and } (z \in o(y).A) \text{ and } (o(z).TO = x))$$

Therefore, $AR[i] \subseteq R[i]$ and $|AR[i]| \leq |R[i]|$ (II)

From (I) and (II) we have that $|AR[i]| \leq |S[i]|$ and thus $c_i \leq 1$.

Furthermore, since the sets $R[i]$ and $S[i]$ have at least one element (i.e. the element $o(i).FROM$) and the set $AR[i]$ will also contain at least the element $o(i).TO$, we have that $|AR[i]| \geq 1$ and $|S[i]| \geq 1$ and thus $c_i > 0$.

However, $0 < c_i \leq 1 \iff 0 \leq (1-c_i) < 1$.

Thus, $0 \leq m_i^{ch}(P) \leq 1, \forall P \subseteq \Theta$

b) Axiom (A.4.5) is satisfied by the function m_i^{ch} since its focals CH_i and $\overline{CH_i}$ are non empty sets by definition:

$$CH_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vc = 1 \right\} \neq \emptyset \text{ and,}$$

$$\overline{CH}_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vc = 0 \right\} \neq \emptyset$$

Thus, the basic probability assignment to the empty set by the function m_i^{ch} is $m_i^{ch}(\emptyset) = 0$

c) Finally, axiom (A.4.6) is satisfied by the function m_i^{ch} since:

$$\sum_{P \subseteq \Theta} m_i^{ch}(P) = m_i^{ch}(CH_i) + m_i^{ch}(\overline{CH}_i) + \sum_{P \subseteq \Theta \ \& \ P \neq CH_i \ \& \ P \neq \overline{CH}_i} m_i^{ch}(P) = c_i + 1 - c_i + 0 = 1$$

□

Theorem 4.2: *The evidence measure*

$$m_i^a(P) = \begin{cases} e_j & \text{if } P = ABS_i \\ 1 - e_j & \text{if } P = \overline{ABS}_i \\ 0 & \text{if } P \subseteq \Theta \ \& \ P \neq ABS_i \ \& \ P \neq \overline{ABS}_i \end{cases}$$

where

$$j = ODC_i$$

$$e_j = \frac{|EXT_s[j]|}{g(EXT[j])} \text{ and,}$$

$$g(EXT[j]) = \begin{cases} |EXT[j]| & \text{if } EXT[j] \neq \emptyset \\ 1 & \text{if } EXT[j] = \emptyset \end{cases}$$

is a basic probability assignment.

PROOF: It is sufficient to prove that axioms (A.4.4), (A.4.5) and (A.4.6) are satisfied by the function m_i^a .

a) Axiom (A.4.4) is satisfied since:

When $P \neq ABS_i$ or $P \neq \overline{ABS}_i$, $m_i^a(P) = 0$ by definition.

When $P = ABS_i$, we have that $m_i^a(ABS_i) = \frac{|EXT_s[j]|}{g(EXT[j])}$ (I).

Two cases are distinguished:

i) $EXT[j] = \emptyset$

According to the definition of the set $EXT_s[j]$ (i.e. definition 4.5), we have that

$$\forall x : x \in EXT_s[j] \rightarrow x \in EXT[j] \text{ and thus } EXT_s[j] \subseteq EXT[j]$$

Therefore,

$$EXT_s[j] = \emptyset$$

and $m_i^a(ABS_i) = 0/1 = 0$

ii) $EXT[j] \neq \emptyset$

Since $EXT_s[j] \subseteq EXT[j]$ it will be also that $|EXT_s[j]| \leq |EXT[j]|$

Therefore, $m_i^a(ABS_i) \leq 1$.

Also, since $|EXT[j]| > 0$ and $EXT_s[j] \geq 0$ it holds that $m_i^a(ABS_i) \geq 0$

So, in any case $0 \leq e_j \leq 1$ and therefore $0 \leq 1 - e_j \leq 1$

Thus $0 \leq m_i^a(\overline{ABS}_i) \leq 1$.

b) Axiom (A.4.5) is satisfied by the function m_i^a since its focals ABS_i and \overline{ABS}_i are non empty sets by definition:

$$ABS_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid va = 1 \right\} \neq \emptyset \text{ and,}$$

$$\overline{ABS}_i \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid va = 0 \right\} \neq \emptyset$$

Thus, the basic probability assignment to the empty set by the function m_i^a is $m_i^a(\emptyset) = 0$

c) Finally, axiom (A.4.6) is satisfied by the function m_i^a since:

$$\sum_{P \subseteq \Theta} m_i^a(P) = m_i^a(ABS_i) + m_i^a(\overline{ABS}_i) + \sum_{P \subseteq \Theta \text{ \& } P \neq ABS_i \text{ \& } P \neq \overline{ABS}_i} m_i^a(P) = e_j + 1 - e_j + 0 = 1$$

□

Theorem 4.3: *The evidence measure*

$$m_{ij}^{ncs}(P) = \begin{cases} d_{ij} & \text{if } P = \overline{M}_j \\ 1 - d_{ij} & \text{if } P = \Theta \\ 0 & \text{if } P \subseteq \Theta \text{ \& } P \neq \overline{M}_j \text{ \& } P \neq \Theta \end{cases}$$

$$\text{where } d_{ij} = \frac{|S[i] - S[j]| + |S[j] - S[i]|}{|S[i] \cup S[j]|}$$

is a basic probability assignment.

PROOF: It is sufficient to prove that axioms (A.4.4), (A.4.5) and (A.4.6) are satisfied by the function m_{ij}^{ncs} .

a) Axiom (A.4.4) is satisfied because if $P \neq \Theta$ and $P \neq \overline{M}_j$ then $m_{ij}^{ncs}(P) = 0$ by definition

It can be easily shown that:

$$|S[i]-S[j]| + |S[j]-S[i]| = |(S[i]-S[j]) \cup (S[j]-S[i])| \quad (I)$$

$$\text{since } (S[i]-S[j]) \cap (S[j]-S[i]) = \emptyset$$

and

$$|(S[i]-S[j]) \cup (S[j]-S[i])| \leq |S[i] \cup S[j]| \quad (II)$$

$$\text{since } ((S[i]-S[j]) \cup (S[j]-S[i])) \subseteq (S[i] \cup S[j]) .$$

(I) and (II) imply that

$$|S[i]-S[j]| + |S[j]-S[i]| \leq |S[i] \cup S[j]|, \text{ thus } d_{ij} \leq 1 \quad (III).$$

Furthermore, $|S[i] \cup S[j]| \geq 1$ (since the sets $S[i]$ and $S[j]$ have at least one element) and

$$|S[i]-S[j]| + |S[j]-S[i]| > 0 . \text{ Thus } d_{ij} \geq 0 \quad (IV) .$$

By (III) and (IV) $0 \leq d_{ij} \leq 1$ and $0 \leq 1-d_{ij} \leq 1$.

b) Axiom (A.4.5) is satisfied by the function m_{ij}^{ncs} since its focals \bar{M}_j and Θ are non empty sets by definition:

$$\bar{M}_j \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vm_j = 0 \right\} \neq \emptyset \text{ and,}$$

$$\Theta \neq \emptyset$$

Thus, the basic probability assignment to the empty set by the function m_{ij}^{ncs} is $m_{ij}^{ncs}(\emptyset) = 0$

c) Finally, axiom (A.4.6) is satisfied by the function m_{ij}^{ncs} since:

$$\sum_{P \subseteq \Theta} m_{ij}^{ncs}(P) = m_{ij}^{ncs}(\bar{M}_j) + m_{ij}^{ncs}(\Theta) + \sum_{P \subseteq \Theta \ \& \ P \neq \bar{M}_j \ \& \ P \neq \Theta} m_{ij}^{ncs}(P) = d_{ij} + 1 - d_{ij} + 0 = 1$$

□

Theorem 4.4: *The evidence measure*

$$m_{ij}^{ncr}(P) = \begin{cases} r_{ij} & \text{if } P = \bar{M}_j \\ 1-r_{ij} & \text{if } P = \Theta \\ 0 & \text{if } P \subseteq \Theta \ \& \ P \neq \bar{M}_j \ \& \ P \neq \Theta \end{cases}$$

$$\text{where } r_{ij} = \frac{|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]|}{|R[i] \cup R[j]|}$$

is a basic probability assignment.

PROOF: It is sufficient to prove that axioms (A.4.4), (A.4.5) and (A.4.6) are satisfied by the function m_{ij}^{ncr} .

a) Axiom (A.4.4) is satisfied because if $P \neq \Theta$ and $P \neq \bar{M}_j$ then $m_{ij}^{ncr}(P) = 0$ by definition.

It can be shown that:

$$|(S[i]-R[i]) \cap R[j]| \leq |R[j]-R[i]| \quad (I) \text{ since}$$

$$((S[i]-R[i]) \cap R[j]) \subseteq R[j] \quad \& \quad ((S[i]-R[i]) \cap R[j]) \cap R[i] = \emptyset$$

and

$$|(S[j]-R[j]) \cap R[i]| \leq |R[i]-R[j]| \quad (II)$$

since

$$((S[j]-R[j]) \cap R[i]) \subseteq R[i] \quad \& \quad ((S[j]-R[j]) \cap R[i]) \cap R[j] = \emptyset .$$

By (I) and (II),

$$|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]| \leq |R[j]-R[i]| + |R[i]-R[j]| \quad (III)$$

Also, since $(R[i]-R[j]) \cap (R[j]-R[i]) = \emptyset$ we have that

$$|R[i]-R[j]| + |R[j]-R[i]| = |(R[i]-R[j]) \cup (R[j]-R[i])| \quad (IV)$$

Furthermore, $(R[i]-R[j]) \cup (R[j]-R[i]) \subseteq (R[i] \cup R[j])$ and thus

$$|(R[i]-R[j]) \cup (R[j]-R[i])| \leq |R[i] \cup R[j]| \quad (V)$$

From (III), (IV) and (V) it follows that

$$|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]| \leq |R[i] \cup R[j]|$$

or, equivalently, that $r_{ij} \leq 1$.

Moreover, $|R[i] \cup R[j]| \geq 1$ (since both $R[i]$ and $R[j]$ have at least one element) and

$$|(S[i]-R[i]) \cap R[j]| + |(S[j]-R[j]) \cap R[i]| \geq 0. \quad \text{Thus, } r_{ij} \geq 0 .$$

Since $0 \leq r_{ij} \leq 1$ also $0 \leq 1-r_{ij} \leq 1$.

b) Axiom (A.4.5) is satisfied by the function m_{ij}^{ncr} since its focals \bar{M}_j and Θ are non empty sets by definition :

$$\bar{M}_j \equiv \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid vm_j = 0 \right\} \neq \emptyset \text{ and,}$$

$$\Theta \neq \emptyset$$

Thus, the basic probability assignment to the empty set by the function m_{ij}^{ncr} is $m_{ij}^{ncr}(\emptyset) = 0$

c) Finally, axiom (A.4.6) is satisfied by the function m_{ij}^{ncr} since:

$$\sum_{P \subseteq \Theta} m_{ij}^{ncr}(P) = m_{ij}^{ncr}(\overline{M}_j) + m_{ij}^{ncr}(\Theta) + \sum_{P \subseteq \Theta \& P \neq \overline{M}_j \& P \neq \Theta} m_{ij}^{ncr}(P) = r_{ij} + 1 - r_{ij} + 0 = 1$$

□

Theorem 4.5: *The basic probability assignments $m_i^{ch}, m_i^a, m_{ij}^{ncs}$ and m_{ij}^{ncr} can be combined into total beliefs in any possible order.*

PROOF: According to theorem 3.4 in ([Sha75] P. 63), it is sufficient to prove that these partial beliefs have cores (i.e. the unions of their focals), whose intersection is not empty. According to the definitions of their focals, their cores are:

$$Cr(m_i^{ch}) \equiv CH_i \cup \overline{CH}_i = \Theta$$

$$Cr(m_i^a) \equiv AB_i \cup \overline{AB}_i = \Theta$$

$$Cr(m_{ij}^{ncs}) = Cr(m_{ij}^{ncr}) \equiv \overline{M}_{ij} \cup \Theta = \Theta, \quad \forall j=1, \dots, n$$

Therefore, their intersection

$$Cr(m_i^{ch}) \cap Cr(m_i^a) \bigcap_{j=1}^n Cr(m_{ij}^{ncs}) \bigcap_{j=1}^n Cr(m_{ij}^{ncr}) \equiv \Theta \neq \emptyset$$

□

Lemma 4.1: *It holds that*
$$\sum_{I \subseteq \{1, 2, \dots, n\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) = 1$$

if $0 \leq x_j \leq 1$ and $\sum_{j=1}^n x_j = 1$

PROOF: This lemma can be proven by induction on n.

For n = 2 we have:

$$\sum_{I \subseteq \{1, 2\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) = (1-x_1)(1-x_2) + x_1(1-x_2) + (1-x_1)x_2 + x_1x_2 = 1$$

We assume that for n = k, it holds that:
$$\sum_{I \subseteq \{1, 2, \dots, k\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) = 1$$

Then for n = k + 1 we can prove it as it follows:

$$\sum_{I \subseteq \{1, 2, \dots, k, k+1\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) =$$

$$\begin{aligned}
& \sum_{I \subseteq \{1,2,\dots,k\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) + \sum_{I \subseteq \{1,2,\dots,k\}} \prod_{j \in I} x_j \prod_{j \in I^c \cup \{k+1\}} (1-x_j) = \\
& \sum_{I \subseteq \{1,2,\dots,k\}} x_{k+1} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) + \sum_{I \subseteq \{1,2,\dots,k\}} (1-x_{k+1}) \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) = \\
& x_{k+1} \sum_{I \subseteq \{1,2,\dots,k\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) + (1-x_{k+1}) \sum_{I \subseteq \{1,2,\dots,k\}} \prod_{j \in I} x_j \prod_{j \in I^c} (1-x_j) = x_{k+1} + 1 - x_{k+1} = 1
\end{aligned}$$

□

Lemma 4.2: *The combination of the basic probability assignments m_{ij}^{ncs} results into the following combined basic probability assignment:*

$$\begin{aligned}
m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j \right) &= \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \text{ and} \\
m^{ncs}(\Theta) &= \prod_{j=1}^n (1-d_{ij})
\end{aligned}$$

PROOF: This theorem can be proven by induction on n.

For n = 2:

The combination of the functions $m_{i1}^{ncs}, m_{i2}^{ncs}$ provides evidence to all the pairwise intersections of their focals $\bar{M}_1 \cap \bar{M}_2 \neq \emptyset, \bar{M}_1 \cap \Theta = \bar{M}_1 \neq \emptyset, \Theta \cap \bar{M}_2 = \bar{M}_2 \neq \emptyset, \Theta \cap \Theta = \Theta \neq \emptyset$

none of which is empty.

Thus, according to the rule of the orthogonal sum (where $k_0 = 0$) we have:

$$\begin{aligned}
m^{ncs}(\bar{M}_1 \cap \bar{M}_2) &= m_{i1}^{ncs}(\bar{M}_1) m_{i2}^{ncs}(\bar{M}_2) = d_{i1} d_{i2} \\
m^{ncs}(\bar{M}_1 \cap \Theta) &= m_{i1}^{ncs}(\bar{M}_1) m_{i2}^{ncs}(\Theta) = d_{i1} (1-d_{i2}) \\
m^{ncs}(\Theta \cap \bar{M}_2) &= m_{i1}^{ncs}(\Theta) m_{i2}^{ncs}(\bar{M}_2) = (1-d_{i1}) d_{i2} \text{ and,} \\
m^{ncs}(\Theta \cap \Theta) &= m_{i1}^{ncs}(\Theta) m_{i2}^{ncs}(\Theta) = (1-d_{i1})(1-d_{i2})
\end{aligned}$$

so the formula to be proven is valid.

Assuming that for n = k it holds that:

$$m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,k\} \& I \neq \emptyset} \bar{M}_j \right) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \text{ and, } m^{ncs}(\Theta) = \prod_{j=1}^k (1-d_{ij})$$

for $n = k + 1$, we have:

The pairwise intersections are:

$$j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset \quad \bar{M}_j \cap \bar{M}_{k+1} = \bigcap_{j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset} \bar{M}_j \neq \emptyset$$

$$j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset \quad \bar{M}_j \cap \Theta = \bigcap_{j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset} \bar{M}_j \neq \emptyset$$

$$\Theta \cap \bar{M}_{k+1} = \bar{M}_{k+1} \neq \emptyset$$

$$\Theta \cap \Theta = \Theta \neq \emptyset$$

Thus, by applying the rule of the orthogonal sum we get:

$$m^{ncs} \left(\bigcap_{j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset} \bar{M}_j \right) = m' \left(\bigcap_{j \in I', I' \subseteq \left\{ \bigcap_{j \in I'} \bar{M}_j \right\} \& I' \neq \emptyset} \bar{M}_j \right) m_{i_{k+1}}^{ncs} (\bar{M}_{k+1}) =$$

$$\prod_{j \in I'} d_{ij} \prod_{j \in I^c} (1-d_{ij}) d_{i_{k+1}} = \prod_{j \in I' \cup \{k+1\}} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m^{ncs} \left(\bigcap_{j \in I, I \subseteq \left\{ \bigcap_{j \in I} \bar{M}_j \right\} \& I \neq \emptyset} \bar{M}_j \right) = m' \left(\bigcap_{j \in I', I' \subseteq \left\{ \bigcap_{j \in I'} \bar{M}_j \right\} \& I' \neq \emptyset} \bar{M}_j \right) m_{i_{k+1}}^{ncs} (\Theta) =$$

$$\prod_{j \in I'} d_{ij} \prod_{j \in I^c} (1-d_{ij}) (1-d_{i_{k+1}}) = \prod_{j \in I'} d_{ij} \prod_{j \in I^c \cup \{k+1\}} (1-d_{i_{k+1}}) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m^{ncs} (\Theta \cap \bar{M}_{k+1}) = m' (\Theta) m_{i_{k+1}}^{ncs} (\bar{M}_{k+1}) = \prod_{j=1}^k (1-d_{ij}) d_{i_{k+1}} = \prod_{j \in \{k+1\}} d_{ij} \prod_{j \in \{1,2,\dots,k\}} (1-d_{ij})$$

$$m^{ncs} (\Theta \cap \Theta) = m' (\Theta) m_{i_{k+1}}^{ncs} (\Theta) = \prod_{j=1}^k (1-d_{ij}) (1-d_{i_{k+1}}) = \prod_{j=1}^{k+1} (1-d_{ij})$$

so the formula to be proven is valid.

□

Lemma 4.3: *The combination of the two basic probability assignments m_{ij}^{ncs} , and m_{ij}^{ncr} results into the simple support function:*

$$m_{ij}^{ncsr}(P) = \begin{cases} 1 - (1-d_{ij})(1-r_{ij}) & \text{if } P = \bar{M}_j \\ (1-d_{ij})(1-r_{ij}) & \text{if } P = \Theta \end{cases}$$

PROOF: The pairwise intersections of the focals \bar{M}_j and Θ of m_{ij}^{ncr} with the focals \bar{M}_j and Θ of m_{ij}^{ncs} are:

$\bar{M}_j \cap \bar{M}_j = \bar{M}_j$, $\bar{M}_j \cap \Theta = \bar{M}_j$, $\Theta \cap \bar{M}_j = \bar{M}_j$ and, $\Theta \cap \Theta = \Theta$. Since none of these intersections is empty the functions m_{ij}^{ncs} and m_{ij}^{ncr} can be combined according to the rule of the orthogonal sum ($k_0 = 0$) as:

$$m_{ij}^{ncsr}(\bar{M}_j) = m_{ij}^{ncr}(\bar{M}_j)m_{ij}^{ncs}(\bar{M}_j) + m_{ij}^{ncr}(\bar{M}_j)m_{ij}^{ncs}(\Theta) + m_{ij}^{ncr}(\Theta)m_{ij}^{ncs}(\bar{M}_j) =$$

$$r_{ij}d_{ij} + r_{ij}(1-d_{ij}) + (1-r_{ij})(1-d_{ij}) = 1 - ((1-r_{ij})(1-d_{ij}))$$

$$\text{Also } m_{ij}^{ncsr}(\Theta) = m_{ij}^{ncr}(\Theta)m_{ij}^{ncs}(\Theta) = (1-r_{ij})(1-d_{ij})$$

□

Lemma 4.4: *The combination of the basic probability assignments m_{ij}^{ncsr} obtained in lemma 4.3 results into the following basic probability assignment:*

$$m^{ncsr}\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\}} \bar{M}_j\right) = \prod_{j \in I} (1 - (1-d_{ij})(1-r_{ij})) \prod_{j \in I^c} (1-d_{ij})(1-r_{ij}) \text{ and,}$$

$$m^{ncsr}(\Theta) = \prod_{j=1}^n (1-d_{ij})(1-r_{ij})$$

PROOF: It can be obtained exactly as the proof of Lemma 4.2 by substituting

m^{ncr} with m^{ncsr}

m_{ij}^{ncr} with m_{ij}^{ncsr} and

d_{ij} with $(1 - (1-d_{ij})(1-r_{ij}))$

□

Theorem 4.6: *a) The combination of the basic probability assignments m_{ij}^{ncs} , $j=1,2,\dots,n$ provides total belief to the non determinance of an attribute i , \overline{DET}_i estimated by:*

$$Bel_1(\overline{DET}_i) = \prod_{j=1}^n d_{ij}$$

b) The combination of the basic probability assignments m_{ij}^{ncr} and m_{ij}^{ncr} , $j=1,2,\dots,n$ provides total belief to the non determinance of an attribute j , \overline{DET}_j estimated by:

$$Bel_2(\overline{DET}_j) = \prod_{j=1}^n (1 - (1-r_{ij})(1-d_{ij}))$$

PROOF:

$$\text{a) } Bel_1(\overline{DET}_i) = Bel_1\left(\bigcap_{j=1}^n \overline{M}_j\right) = \sum_{X \subseteq \bigcap_{j=1}^n \overline{M}_j} m^{ncsr}(X) = m^{ncsr}\left(\bigcap_{j=1}^n \overline{M}_j\right)$$

By lemma 4.2,

$$Bel_1(\overline{DET}_i) = \prod_{j=1}^n d_{ij}$$

$$\text{b) } Bel_2(\overline{DET}_i) = Bel_2\left(\bigcap_{j=1}^n \overline{M}_j\right) = \sum_{X \subseteq \bigcap_{j=1}^n \overline{M}_j} m^{ncsr}(X) = m^{ncsr}\left(\bigcap_{j=1}^n \overline{M}_j\right)$$

By lemma 4.3,

$$Bel_2(\overline{DET}_i) = \prod_{j=1}^n (1 - (1 - r_{ij})(1 - d_{ij})) .$$

□

Theorem 4.7: *The combination of the basic probability assignments m_i^a, m_i^{ch} , and m^{ncsr} results into the following combined beliefs:*

$$Bel(DOM_i) = c_i e_u$$

$$Bel(\overline{DOM}_i) = (1 - c_i e_u) \prod_{j=1}^n d_{ij}$$

$$P^*(DOM_i) = 1 - (1 - c_i e_u) \prod_{j=1}^n d_{ij}$$

where $u = ODC_i$.

PROOF: By theorem 4.5, we can combine these basic probability assignments in any possible order. Thus, we will first combine the m_i^a , and m_i^{ch} .

Step 1: The pairwise intersections of the focals of the m_i^a , and m_i^{ch} are:

$$ABS_i \cap CH_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \right\} \neq \emptyset$$

Consequently, the application of the rule of the orthogonal sum gives($k_0 = 0$):

$$m'(ABS_i \cap CH_i) = m_i^a(ABS_i) m_i^{ch}(CH_i) = e_u c_i$$

$$m'(\overline{ABS}_i \cap CH_i) = m_i^a(\overline{ABS}_i) m_i^{ch}(CH_i) = (1 - e_u) c_i$$

$$m'(ABS_i \cap \overline{CH}_i) = m_i^a(ABS_i) m_i^{ch}(\overline{CH}_i) = e_u (1 - c_i) \text{ and}$$

$$m'(\overline{ABS}_i \cap \overline{CH}_i) = m_i^a(\overline{ABS}_i) m_i^{ch}(\overline{CH}_i) = (1 - e_u) (1 - c_i)$$

Step 2: By lemma 4.2, we have that the focals of the combination of the basic probability assignment m^{ncs} are the sets $\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j$ and the set \emptyset . The intersections of these

focals with the focals of the m' basic probability assignment, obtained in the previous step, are:

$$ABS_i \cap CH_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\forall j \in I : vm_j = 0) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \overline{M}_j = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\forall j \in I : vm_j = 0) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\forall j \in I : vm_j = 0) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j =$$

$$\left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\forall j \in I : vm_j = 0) \right\} \neq \emptyset$$

$$ABS_i \cap CH_i \cap \emptyset = ABS_i \cap CH_i \neq \emptyset \text{ and } \overline{ABS}_i \cap CH_i \cap \emptyset = \overline{ABS}_i \cap CH_i \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \emptyset = ABS_i \cap \overline{CH}_i \neq \emptyset \text{ and } \overline{ABS}_i \cap \overline{CH}_i \cap \emptyset = \overline{ABS}_i \cap \overline{CH}_i \neq \emptyset$$

Therefore, the application of the rule of the orthogonal sum ($k_0 = 0$) gives:

$$m''(ABS_i \cap CH_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j) = m'(ABS_i \cap CH_i) m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) = c_i e_u \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j) = m'(\overline{ABS}_i \cap CH_i) m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) =$$

$$c_i (1 - e_u) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$\begin{aligned}
& m''(ABS_i \cap \overline{CH}_i \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\}, I \neq \emptyset} \overline{M}_j) = & m'(ABS_i \cap \overline{CH}_i) m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\}, I \neq \emptyset} \overline{M}_j \right) = \\
& (1-c_i) e_u \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) & \\
& m''(\overline{ABS}_i \cap \overline{CH}_i \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\}, I \neq \emptyset} \overline{M}_j) = & m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\}, I \neq \emptyset} \overline{M}_j \right) = \\
& (1-c_i)(1-e_u) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) & \\
& m''(ABS_i \cap CH_i \cap \Theta) = m'(ABS_i \cap CH_i) m^{ncs}(\Theta) = c_i e_u \prod_{j=1}^n (1-d_{ij}) & \\
& m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m^{ncs}(\Theta) = c_i (1-e_u) \prod_{j=1}^n (1-d_{ij}) & \\
& m''(ABS_i \cap \overline{CH}_i \cap \Theta) = m'(ABS_i \cap \overline{CH}_i) m^{ncs}(\Theta) = (1-c_i) e_u \prod_{j=1}^n (1-d_{ij}) & \\
& m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncs}(\Theta) = (1-c_i)(1-e_u) \prod_{j=1}^n (1-d_{ij}) &
\end{aligned}$$

Then by accumulating these basic probability assignments into beliefs according to formula (4.1) we get:

$$\begin{aligned}
& Bel(DOM_i) = Bel((ABS_i \cap CH_i) \cup DET_i) = \sum_{X \subseteq (ABS_i \cap CH_i) \cup DET_i} m''(X) = \\
& \sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} m''(ABS_i \cap CH_i \bigcap_{j \in I} \overline{M}_j) + m''(ABS_i \cap CH_i \cap \Theta) = \\
& c_i e_u \left(\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \left(\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \right) + \prod_{j=1}^n (1-d_{ij}) \right)
\end{aligned}$$

$$\text{However: } \sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \left(\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \right) + \prod_{j=1}^n (1-d_{ij}) = \sum_{I \subseteq \{1,2,\dots,n\}} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = 1$$

by lemma 4.1

Thus, $Bel(DOM_i) = c_i e_u$

Also,

$$\begin{aligned}
& Bel(\overline{DOM}_i) = Bel(\overline{(ABS_i \cap CH_i) \cup DET_i}) = Bel(\overline{(ABS_i \cap CH_i)} \cap \overline{DET_i}) = \\
& Bel(\overline{(ABS_i \cap CH_i)} \bigcap_{j=1}^n \overline{M}_j) = \sum_{X \subseteq \overline{(ABS_i \cap CH_i)} \cap \overline{DET_i}} m(X) =
\end{aligned}$$

$$m''((\overline{ABS}_i \cup CH_i) \bigcap_{j=1}^n \overline{M}_j) + m''((ABS_i \cup \overline{CH}_i) \bigcap_{j=1}^n \overline{M}_j) + m''((\overline{ABS}_i \cup \overline{CH}_i) \bigcap_{j=1}^n \overline{M}_j) =$$

$$(1-e_u)c_i \prod_{j=1}^n d_{ij} + e_u(1-c_i) \prod_{j=1}^n d_{ij} + (1-e_u)(1-c_i) \prod_{j=1}^n d_{ij} = (1-c_i e_u) \prod_{j=1}^n d_{ij}$$

$$\text{Therefore, } P^*(DOM_i) = 1 - (1-c_i e_u) \prod_{j=1}^n d_{ij}$$

□

Theorem 4.8: The combination of the basic probability assignments m_i^a, m_i^{ch} with the basic probability assignments m_{ij}^{ncs} and m_{ij}^{ncr} results into the following combined beliefs:

$$Bel(DOM_i) = c_i e_u$$

$$Bel(\overline{DOM}_i) = (1-c_i e_u) \prod_{j=1}^n (1-(1-r_{ij})(1-d_{ij}))$$

$$P^*(DOM_i) = 1 - (1-c_i e_u) \prod_{j=1}^n (1-(1-r_{ij})(1-d_{ij}))$$

where $u = ODC_i$.

PROOF: As it shown in *theorem 4.7*, the basic probability assignments m_i^{ch} and m_i^a , can be combined providing belief to the following focals:

$$m'(ABS_i \bigcap CH_i) = m_i^a(ABS_i) m_i^{ch}(CH_i) = e_u c_i$$

$$m'(\overline{ABS}_i \bigcap CH_i) = m_i^a(\overline{ABS}_i) m_i^{ch}(CH_i) = (1-e_u) c_i$$

$$m'(ABS_i \bigcap \overline{CH}_i) = m_i^a(ABS_i) m_i^{ch}(\overline{CH}_i) = e_u (1-c_i) \text{ and}$$

$$m'(\overline{ABS}_i \bigcap \overline{CH}_i) = m_i^a(\overline{ABS}_i) m_i^{ch}(\overline{CH}_i) = (1-e_u)(1-c_i)$$

In *theorem 4.7* it has also shown that the intersections of the focals of the basic probability assignment m^{ncsr} , which in turn are the sets $\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j$ and the set \emptyset , with the

focals of the m' basic probability assignments are non empty sets. Therefore these functions can be combined according to the rule of the orthogonal sum ($k_0 = 0$) as follows:

$$m''(ABS_i \bigcap CH_i \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) = m'(ABS_i \bigcap CH_i) m^{ncsr}(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) =$$

$$c_i e_u \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \cap \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) = m'(\overline{ABS}_i \cap CH_i) m^{ncsr}(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) =$$

$$c_i(1-e_u) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) = m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncsr}(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) =$$

$$(1-c_i)e_u \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) = m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncsr}(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j) =$$

$$(1-c_i)(1-e_u) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m^{ncsr}(\Theta) = c_i e_u \prod_{j=1}^n (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m^{ncsr}(\Theta) = c_i (1-e_u) \prod_{j=1}^n (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncsr}(\Theta) = (1-c_i) e_u \prod_{j=1}^n (1-d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m^{ncsr}(\Theta) = (1-c_i)(1-e_u) \prod_{j=1}^n (1-d_{ij})$$

Then by accumulating these basic probability assignments into beliefs according to formula (4.1) we get:

$$Bel(DOM_i) = Bel((\overline{ABS}_i \cap CH_i) \cup DET_i) = \sum_{X \subseteq (\overline{ABS}_i \cap CH_i) \cup DET_i} m''(X) =$$

$$\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} m''(\overline{ABS}_i \cap CH_i \cap \bigcap_{j \in I} \overline{M}_j) + m''(\overline{ABS}_i \cap CH_i \cap \Theta) =$$

$$c_i e_u \left(\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \left(\prod_{j \in I} (1-(1-d_{ij})(1-r_{ij})) \right) \prod_{j \in I^c} (1-d_{ij})(1-r_{ij}) + \prod_{j=1}^n (1-d_{ij})(1-r_{ij}) \right)$$

However:

$$\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \left(\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-(1-d_{ij})(1-r_{ij})) \right)$$

$$+ \prod_{j=1}^n ((1-d_{ij})(1-r_{ij})) = \sum_{I \subseteq \{1,2,\dots,n\}} \left(\prod_{j \in I} (1-(1-d_{ij})(1-r_{ij})) \prod_{j \in I^c} ((1-d_{ij})(1-r_{ij})) \right) = 1$$

by lemma 4.1

Thus,

$$Bel(DOM_i) = c_i e_u$$

and

$$Bel(\overline{DOM}_i) = Bel(\overline{(ABS_i \cap CH_i) \cup DET_i}) = Bel(\overline{(ABS_i \cup \overline{CH_i}) \cap \overline{DET_i}}) = Bel(\overline{(ABS_i \cup \overline{CH_i}) \cap \bigcap_{j=1}^n \overline{M_j}}) =$$

$$\sum_{X \subseteq \overline{(ABS_i \cup \overline{CH_i}) \cap \overline{DET_i}}} m(X) = m''(\overline{(ABS_i \cup \overline{CH_i}) \cap \bigcap_{j=1}^n \overline{M_j}}) + m''((ABS_i \cup \overline{CH_i}) \cap \bigcap_{j=1}^n \overline{M_j}) + m''(\overline{(ABS_i \cup \overline{CH_i}) \cap \bigcap_{j=1}^n \overline{M_j}}) =$$

$$(1 - e_u) c_i \prod_{j=1}^n (1 - (1 - d_{ij})(1 - r_{ij})) + e_u (1 - c_i) \prod_{j=1}^n (1 - (1 - d_{ij})(1 - r_{ij})) +$$

$$(1 - e_u)(1 - c_i) \prod_{j=1}^n (1 - (1 - d_{ij})(1 - r_{ij})) = (1 - c_i e_u) \prod_{j=1}^n (1 - (1 - d_{ij})(1 - r_{ij}))$$

$$\text{Therefore, } P^*(DOM_i) = 1 - (1 - c_i e_u) \prod_{j=1}^n (1 - (1 - d_{ij})(1 - r_{ij})) .$$

□

Theorem 4.9: *The combination of any simple support basic probability assignment to the determinance of an attribute i , defined as:*

$$s(P) = \begin{cases} b & \text{if } P = \bigcup_{j=1}^n M_j \\ 1 - b & \text{if } P = \Theta \end{cases}$$

where $0 \leq b \leq 1$ can be combined with the basic probability assignments m_i^{ch} , m_i^a and, m_{ij}^{ncs} only if:

$$b \prod_{j=1}^n d_{ij} < 1$$

Their combination, when valid, results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = e_u c_i + (1 - e_u c_i) \frac{b(1 - \prod_{j=1}^n d_{ij})}{1 - b \prod_{j=1}^n d_{ij}}$$

$$P^*(DOM_i) = 1 - (1 - e_u c_i) \prod_{j=1}^n d_{ij} \frac{1 - b}{1 - b \prod_{j=1}^n d_{ij}}$$

where $u = ODC_i$.

PROOF: The core of the function s , $Cr(s)$, is the set Θ ($Cr(s) \equiv \bigcup_{j=1}^n M_j \cup \Theta = \Theta$). The intersection of this core with the cores of the basic probability assignments m_i^{ch} , m_i^a , m_{ij}^{ncs} , ($j=1, \dots, n$), which are all equal to Θ , as it shown in *theorem 4.5*, also equals Θ . Since Θ is a non empty set, these functions are combinable in any possible order according to the *theorem 3.4* in ([Sha75], P. 63).

Step 1: First, function s is combined with the functions m_{ij}^{ncs} . As it shown in *lemma 4.2*, the combination of these latter functions, results in the following basic probability assignment:

$$m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j \right) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \text{ and}$$

$$m^{ncs}(\Theta) = \prod_{j=1}^n (1-d_{ij})$$

The pairwise intersections of the focals of the m^{ncs} with the focals of the s function are:

$$\left(\bigcap_{j=1}^n \bar{M}_j \right) \cap \left(\bigcup_{j=1}^n M_j \right) = \emptyset$$

$$\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j \right) \cap \left(\bigcup_{j=1}^n M_j \right) \neq \emptyset$$

$$\Theta \cap \left(\bigcup_{j=1}^n M_j \right) = \bigcup_{j=1}^n M_j \neq \emptyset$$

$$\left(\bigcap_{j=1}^n \bar{M}_j \right) \cap \Theta = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j \right) \cap \Theta = \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j \neq \emptyset$$

$$\Theta \cap \Theta = \Theta \neq \emptyset$$

Thus, the function s can be combined with the function m^{ncs} , only if (see *theorem 3.1* in [Sha75] P. 60):

$$k_0 = \sum_{A_p \cap B_q = \emptyset \& A_p, B_q \subseteq \Theta} s(A_p) m^{ncs}(B_q) = s \left(\bigcup_{j=1}^n M_j \right) m^{ncs} \left(\bigcap_{j=1}^n \bar{M}_j \right) = b \prod_{j=1}^n d_{ij} < 1$$

This condition ensures the well-definedness of the rule of the orthogonal sum, whose application gives:

$$m\left(\left(\bigcap_{j \in I, I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j\right) \cap \left(\bigcup_{j=1}^n M_j\right)\right) = m^{ncs}\left(\left(\bigcap_{j \in I, I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j\right) s\left(\bigcup_{j=1}^n M_j\right)\right) = \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})b}{1-k_0}$$

$$m\left(\Theta \cap \left(\bigcup_{j=1}^n M_j\right)\right) = m^{ncs}\left(\Theta\right) s\left(\bigcup_{j=1}^n M_j\right) = \frac{\prod_{j=1}^n (1-d_{ij})b}{1-k_0}$$

$$m\left(\left(\bigcap_{j=1}^n \bar{M}_j\right) \cap \Theta\right) = m^{ncs}\left(\left(\bigcap_{j=1}^n \bar{M}_j\right) s(\Theta)\right) = \frac{\prod_{j=1}^n d_{ij}(1-b)}{1-k_0}$$

$$m\left(\left(\bigcap_{j \in I, I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j\right) \cap \Theta\right) = m^{ncs}\left(\left(\bigcap_{j \in I, I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j\right) s(\Theta)\right) = \frac{\prod_{j=1}^n (1-d_{ij})(1-b)}{1-k_0}$$

$$m(\Theta \cap \Theta) = m^{ncs}(\Theta) s(\Theta) = \frac{\prod_{j=1}^n (1-d_{ij})(1-b)}{1-k_0}$$

Next, these belief estimates are accumulated into belief to DET_i and \overline{DET}_i , according to the rule $Bel(A) = \sum_{B \subseteq A} m(B)$ as it follows:

$$\begin{aligned} Bel(DET_i) &= \sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} m\left(\left(\bigcap_{j \in I, I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \bar{M}_j\right) \cap \left(\bigcup_{j=1}^n M_j\right)\right) + m\left(\Theta \cap \left(\bigcup_{j=1}^n M_j\right)\right) \\ &= \frac{\sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})b + \prod_{j=1}^n (1-d_{ij})b}{1-k_0} \end{aligned}$$

However, by *lemma 4.1*:

$$\sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = 1 - \prod_{j=1}^n d_{ij} - \prod_{j=1}^n (1-d_{ij})$$

Therefore,

$$Bel(DET_i) = b \frac{1 - \prod_{j=1}^n d_{ij}}{1-k_0}$$

Also,

$$Bel(\overline{DET}_i) = m((\bigcap_{j=1}^n \overline{M}_j) \cap \Theta) = m^{ncs}(\bigcap_{j=1}^n \overline{M}_j) s(\Theta) = \prod_{j=1}^n d_{ij} \frac{(1-b)}{(1-k_0)}$$

Step 2: As it has been proven in the *theorem 4.7*, the combination of the basic probability assignments m_i^{ch} and m_i^a results in the basic probability assignment m' , having the following focals:

$$ABS_i \cap CH_i \quad ABS_i \cap \overline{CH}_i \quad \overline{ABS}_i \cap CH_i \quad \overline{ABS}_i \cap \overline{CH}_i$$

Also, from the focals of the combination of function s with m^{ncs} , the only focals which are not subsets of either DET_i or \overline{DET}_i are Θ and the sets $\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j$. Let us call these

$$j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset$$

sets S_I . Then, the pairwise intersections of the focals of m' and m are:

$$ABS_i \cap CH_i \cap DET_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\exists j : (vm_j = 1)) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap DET_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\exists j : (vm_j = 1)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap DET_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\exists j : (vm_j = 1)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap DET_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\exists j : (vm_j = 1)) \right\} \neq \emptyset$$

$$ABS_i \cap CH_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\forall j : (vm_j = 0)) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\forall j : (vm_j = 0)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\forall j : (vm_j = 0)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\forall j : (vm_j = 0)) \right\} \neq \emptyset$$

$$ABS_i \cap CH_i \cap S_I = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\forall I : I \subset \{1, 2, \dots, n\} \& I \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap S_I = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap S_I = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap S_I = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$ABS_i \cap CH_i \cap \Theta \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \Theta \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap \Theta \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap \Theta \neq \emptyset$$

Therefore, the application of the rule of the orthogonal sum gives (m is the basic probability assignment obtained in step 1):

$$m''(ABS_i \cap CH_i \cap \overline{DET}_i) = m'(ABS_i \cap CH_i) m(\overline{DET}_i) = c_i e_u \prod_{j=1}^n d_{ij} \frac{(1-b)}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap CH_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap CH_i) m(\overline{DET}_i) = c_i (1-e_u) \prod_{j=1}^n d_{ij} \frac{(1-b)}{(1-k_0)}$$

$$m''(ABS_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(ABS_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i) e_u \prod_{j=1}^n d_{ij} \frac{(1-b)}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i)(1-e_u) \prod_{j=1}^n d_{ij} \frac{(1-b)}{(1-k_0)}$$

$$m''(ABS_i \cap CH_i \cap DET_i) = m'(ABS_i \cap CH_i) m(DET_i) = c_i e_u b \frac{(1 - \prod_{j=1}^n d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap CH_i \cap DET_i) = m'(\overline{ABS}_i \cap CH_i) m(DET_i) = c_i (1-e_u) b \frac{(1 - \prod_{j=1}^n d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i) e_u b \frac{(1-\prod_{j=1}^n d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap DET_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(DET_i) = (1-c_i)(1-e_u) b \frac{(1-\prod_{j=1}^n d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap CH_i \cap S_I) = m'(\overline{ABS}_i \cap CH_i) m(S_I) = c_i e_u (1-b) \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})}{(1-k_0)}$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$m''(\overline{ABS}_i \cap CH_i \cap S_I) = m'(\overline{ABS}_i \cap CH_i) m(S_I) = c_i (1-e_u) (1-b) \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})}{(1-k_0)}$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap S_I) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(S_I) = (1-c_i) e_u (1-b) \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})}{(1-k_0)}$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap S_I) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(S_I) = (1-c_i)(1-e_u)(1-b) \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})}{(1-k_0)}$$

$$\forall I : I \subset \{1, 2, \dots, n\} \ \& \ I \neq \emptyset$$

$$m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m(\Theta) = c_i e_u \frac{\prod_{j=1}^n (1-d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m(\Theta) = c_i (1-e_u) \frac{(1-b) \prod_{j=1}^n (1-d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\Theta) = (1-c_i) e_u \frac{(1-b) \prod_{j=1}^n (1-d_{ij})}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\Theta) = (1-c_i)(1-e_u) \frac{(1-b) \prod_{j=1}^n (1-d_{ij})}{(1-k_0)}$$

Then by accumulating these basic probability assignments into total beliefs according to the formula (4.1) we get:

$$\begin{aligned}
Bel(DOM_i) &= \sum_{X \subseteq (ABS_i \cap CH_i) \cup DET_i} m''(X) = m''((ABS_i \cap CH_i) \cap DET_i) + \\
& m''((ABS_i \cap CH_i) \cap \overline{DET_i}) + m''((ABS_i \cap CH_i) \cap \Theta) + \sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} m''((ABS_i \cap CH_i) \cap S_I) = \\
& b \frac{\prod_{j=1}^n (1-d_{ij})}{(1-k_0)} (c_i e_u + c_i (1-e_u) + (1-c_i) e_u + (1-c_i) (1-e_u)) + \\
& c_i e_u (1-b) \left(\prod_{j=1}^n (d_{ij}) + \prod_{j=1}^n (1-d_{ij}) + \sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})}{(1-k_0)} \right)
\end{aligned}$$

However, by *lemma 4.1*

$$\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = 1 - \prod_{j=1}^n d_{ij} - \prod_{j=1}^n (1-d_{ij})$$

Thus,

$$Bel(DOM_i) = e_u c_i + (1-e_u c_i) \frac{b \prod_{j=1}^n (1-d_{ij})}{(1-k_0)} = e_u c_i + (1-e_u c_i) \frac{b \prod_{j=1}^n (1-d_{ij})}{(1-b \prod_{j=1}^n d_{ij})}$$

Also,

$$\begin{aligned}
Bel(\overline{DOM_i}) &= \sum_{X \subseteq (ABS_i \cap CH_i) \cup DET_i} m''(X) = \sum_{X \subseteq (ABS_i \cup CH_i) \cap \overline{DET_i}} m''(X) \\
& m''((ABS_i \cap CH_i) \cap \overline{DET_i}) + m''((ABS_i \cap \overline{CH_i}) \cap \overline{DET_i}) + m''((\overline{ABS_i} \cap \overline{CH_i}) \cap \overline{DET_i}) = \\
& \frac{(1-b) \prod_{j=1}^n d_{ij}}{(1-k_0)} ((1-e_u) c_i + e_u (1-c_i) + (1-e_u) (1-c_i)) = (1-c_i e_u) \frac{(1-b) \prod_{j=1}^n d_{ij}}{(1-b \prod_{j=1}^n d_{ij})}
\end{aligned}$$

$$\text{Therefore, } P^*(DOM_i) = 1 - (1-c_i e_u) \frac{(1-b) \prod_{j=1}^n d_{ij}}{(1-b \prod_{j=1}^n d_{ij})}.$$

□

Theorem 4.10: *The combination of any simple support basic probability assignment to the determinance of an attribute i combinable with the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} will result in belief and plausibility measures of DOM_i , $Bel'(DOM_i)$ and $P'*(DOM_i)$ such that:*

$$Bel'(DOM_i) \geq Bel(DOM_i)$$

$$P'*(DOM_i) \geq P*(DOM_i)$$

where $Bel(DOM_i), P*(DOM_i)$ are the belief and the plausibility of DOM_i estimated from the combination of only the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} .

PROOF: By theorems 4.7 and 4.9, we have that:

$$Bel'(DOM_i) = Bel(DOM_i) + (1 - e_u c_i) \frac{b \prod_{j=1}^n (1 - d_{ij})}{(1 - b \prod_{j=1}^n d_{ij})}$$

$$\text{However, since } 0 \leq e_u, c_i, b, d_{ij} \leq 1 \text{ we have } (1 - e_u c_i) \frac{b \prod_{j=1}^n (1 - d_{ij})}{(1 - b \prod_{j=1}^n d_{ij})} \geq 0$$

Thus, $Bel'(DOM_i) \geq Bel(DOM_i)$

Also, according to the same theorems, it holds that:

$$Bel'(\overline{DOM}_i) = Bel(\overline{DOM}_i) \frac{(1-b)}{(1-b \prod_{j=1}^n d_{ij})}$$

However, $\prod_{j=1}^n d_{ij} \leq 1$ since $d_{ij} \leq 1 \forall j$. Therefore, $\frac{(1-b)}{(1-b \prod_{j=1}^n d_{ij})} \leq 1$ and hence:

$$Bel'(\overline{DOM}_i) \leq Bel(\overline{DOM}_i) \rightarrow P'*(DOM_i) \geq P*(DOM_i).$$

□

Theorem 4.11: *The combination of any simple support basic probability assignment to the non determinance of an attribute i , defined as:*

$$s(P) = \begin{cases} b & \text{if } P = \bigcap_{j=1}^n \overline{M}_j \\ 1-b & \text{if } P = \Theta \end{cases}$$

where $0 \leq b \leq 1$ with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = e_u c_i$$

$$P^*(DOM_i) = 1 - (1 - e_u c_i) \left(\prod_{j=1}^n d_{ij} + b \prod_{j=1}^n (1 - d_{ij}) \right)$$

where $u = ODC_i$

PROOF: The core of the function s , $Cr(s)$, is the set Θ ($Cr(s) \equiv \bigcap_{j=1}^n M_j \cup \Theta = \Theta$). Thus its intersection with the cores of the basic probability assignments $m_i^{ch}, m_i^a, m_{ij}^{ncs}$, ($j=1, \dots, n$), which are all equal to Θ , as shown in *theorem 4.5*, also equals Θ . Since Θ is a non empty set, these functions are combinable in any possible order according to *theorem 3.4* in ([Sha75], P. 63).

Step 1: First, s is combined with the functions m_{ij}^{ncs} . As shown in *lemma 4.2*, the combination of these latter functions, results into the following basic probability assignment:

$$m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \right) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij}) \quad \text{and} \quad m^{ncs}(\Theta) = \prod_{j=1}^n (1 - d_{ij})$$

The pairwise intersections of the focals of the m^{ncs} with the focals of the s function are:

$$\left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \right) \cap \left(\bigcap_{j=1}^n \bar{M}_j \right) = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$\Theta \cap \left(\bigcap_{j=1}^n \bar{M}_j \right) = \bigcup_{j=1}^n M_j = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$\left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \right) \cap \Theta = \bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \neq \emptyset$$

$$\Theta \cap \Theta = \Theta \neq \emptyset$$

Thus, the function s can be combined with the function m^{ncs} , according to the rule of the orthogonal sum ($k_0 = 0$):

$$m \left(\left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \right) \cap \left(\bigcap_{j=1}^n \bar{M}_j \right) \right) = m^{ncs} \left(\bigcap_{j \in I, I \subseteq \{1, 2, \dots, n\}} \bar{M}_j \right) s \left(\bigcap_{j=1}^n \bar{M}_j \right) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij}) b$$

$$m \left(\Theta \cap \left(\bigcap_{j=1}^n \bar{M}_j \right) \right) = m^{ncs}(\Theta) s \left(\bigcap_{j=1}^n \bar{M}_j \right) = \prod_{j=1}^n (1 - d_{ij}) b$$

$$m\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j\right) \cap \Theta = m^{ncs}\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j\right) s(\Theta) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})(1-b)$$

$$m(\Theta \cap \Theta) = m^{ncs}(\Theta) s(\Theta) = \prod_{j=1}^n (1-d_{ij})(1-b)$$

Next, these belief estimates are accumulated into belief to \overline{DET}_i , according to the rule

$$Bel(A) = \sum_{B \subseteq A} m(B) \text{ as follows:}$$

$$Bel(\overline{DET}_i) =$$

$$m\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j\right) \cap \left(\bigcap_{j=1}^n \overline{M}_j\right) + m\left(\Theta \cap \left(\bigcap_{j=1}^n \overline{M}_j\right)\right) + m\left(\bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j\right) \cap \Theta =$$

$$\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} b \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) + b \prod_{j=1}^n (1-d_{ij}) + (1-b) \prod_{j=1}^n d_{ij}$$

However, according to *lemma 4.1*, we have that:

$$\sum_{I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = (1 - \prod_{j=1}^n (1-d_{ij}))$$

Thus,

$$Bel(\overline{DET}_i) = \prod_{j=1}^n d_{ij} + b(1 - \prod_{j=1}^n d_{ij})$$

$$Bel(\Theta) = (1-b) \prod_{j=1}^n (1-d_{ij})$$

Step 2: As shown in *theorem 4.7*, the combination of the basic probability assignments m_i^{ch} and m_i^a results into the basic probability assignment m' , providing degrees of support to the following focals:

$$ABS_i \cap CH_i \quad ABS_i \cap \overline{CH}_i$$

$$\overline{ABS}_i \cap CH_i \quad \overline{ABS}_i \cap \overline{CH}_i$$

Since the focals of m are the sets:

$$\overline{DET}_i, \quad \bigcap_{j \in I, I \subseteq \{1,2,\dots,n\} \& I \neq \emptyset} \overline{M}_j \text{ and } \Theta$$

their pairwise intersections with the focals of m' are:

$$ABS_i \cap CH_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\forall j : (vm_j = 0)) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\forall j : (vm_j = 1)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\forall j : (vm_j = 1)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i = \left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\forall j : (vm_j = 1)) \right\} \neq \emptyset$$

$$ABS_i \cap CH_i \cap \left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) =$$

$$\left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 1) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap \left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) =$$

$$\left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 1) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) =$$

$$\left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 1) \text{ and } (vc = 0) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap \left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \& I \neq \emptyset} \overline{M}_j \right) =$$

$$\left\{ [vc, va, vm_1, vm_2, \dots, vm_n] \mid (va = 0) \text{ and } (vc = 0) \text{ and } (\forall j \in I : (vm_j = 0)) \right\} \neq \emptyset$$

$$ABS_i \cap CH_i \cap \Theta = ABS_i \cap CH_i \neq \emptyset$$

$$\overline{ABS}_i \cap CH_i \cap \Theta = \overline{ABS}_i \cap CH_i \neq \emptyset$$

$$ABS_i \cap \overline{CH}_i \cap \Theta = ABS_i \cap \overline{CH}_i \neq \emptyset$$

$$\overline{ABS}_i \cap \overline{CH}_i \cap \Theta = \overline{ABS}_i \cap \overline{CH}_i \neq \emptyset$$

Therefore, the application of the rule of the orthogonal sum gives:

$$m''(ABS_i \cap CH_i \cap \overline{DET}_i) = m'(ABS_i \cap CH_i) m(\overline{DET}_i) = c_i e_u \left(\prod_{j=1}^n d_{ij} + b(1 - \prod_{j=1}^n d_{ij}) \right)$$

$$m''(\overline{ABS}_i \cap CH_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap CH_i) m(\overline{DET}_i) = c_i (1 - e_u) \left(\prod_{j=1}^n d_{ij} + b(1 - \prod_{j=1}^n d_{ij}) \right)$$

$$m''(ABS_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(ABS_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1 - c_i) e_u \left(\prod_{j=1}^n d_{ij} + b(1 - \prod_{j=1}^n d_{ij}) \right)$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1 - c_i)(1 - e_u) \left(\prod_{j=1}^n d_{ij} + b(1 - \prod_{j=1}^n d_{ij}) \right)$$

$$m''(ABS_i \cap CH_i \cap \bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) = m'(ABS_i \cap CH_i) m(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$c_i e_u (1 - b) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \cap \bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$m'(\overline{ABS}_i \cap CH_i) m(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$c_i (1 - e_u) (1 - b) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$m''(ABS_i \cap \overline{CH}_i \cap \bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$m'(ABS_i \cap \overline{CH}_i) m(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$(1 - c_i) e_u (1 - b) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$m'(\overline{ABS}_i \cap \overline{CH}_i) m(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \overline{M}_j) =$$

$$(1 - c_i)(1 - e_u)(1 - b) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1 - d_{ij})$$

$$m''(ABS_i \cap CH_i \cap \Theta) = m'(ABS_i \cap CH_i) m(\Theta) = c_i e_u (1 - b) \prod_{j=1}^n (1 - d_{ij})$$

$$m''(\overline{ABS}_i \cap CH_i \cap \Theta) = m'(\overline{ABS}_i \cap CH_i) m(\Theta) = c_i (1 - e_u) (1 - b) \prod_{j=1}^n (1 - d_{ij})$$

$$m''(ABS_i \cap \overline{CH}_i \cap \Theta) = m'(ABS_i \cap \overline{CH}_i) m(\Theta) = (1 - c_i) e_u (1 - b) \prod_{j=1}^n (1 - d_{ij})$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \Theta) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\Theta) = (1 - c_i)(1 - e_u)(1 - b) \prod_{j=1}^n (1 - d_{ij})$$

Then by accumulating these basic probability assignments into total beliefs according to the formula (4.1) we get:

$$\begin{aligned}
Bel(DOM_i) &= \sum_{X \subseteq (ABS_i \cap CH_i) \cup \overline{DET}_i} m''(X) = \\
& m''((ABS_i \cap CH_i) \cap \overline{DET}_i) + \sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} m''((ABS_i \cap CH_i) \cap (\bigcap_{j \in I} \overline{M}_j)) + m''((ABS_i \cap CH_i) \cap \Theta) = \\
& c_i e_u \left(\prod_{j=1}^n d_{ij} + b \left(1 - \prod_{j=1}^n d_{ij} \right) \right) + c_i e_u \sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} (1-b) \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) + c_i e_u (1-b) \prod_{j=1}^n (1-d_{ij})
\end{aligned}$$

However, since by *lemma 4.1*

$$\sum_{I \subset \{1,2,\dots,n\} \& I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = 1 - \prod_{j=1}^n d_{ij} - \prod_{j=1}^n (1-d_{ij})$$

$$Bel(DOM_i) = e_u c_i$$

$$\begin{aligned}
Bel(\overline{DOM}_i) &= \sum_{X \subseteq (\overline{ABS}_i \cup \overline{CH}_i) \cap \overline{DET}_i} m''(X) = \\
& m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) + m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) + m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = \\
& ((1-e_u)c_i) + e_u(1-c_i) + (1-e_u)(1-c_i) \left(\prod_{j=1}^n d_{ij} + b \left(1 - \prod_{j=1}^n d_{ij} \right) \right) = (1-c_i e_u) \left(\prod_{j=1}^n d_{ij} + b \left(1 - \prod_{j=1}^n d_{ij} \right) \right)
\end{aligned}$$

Therefore, $P^*(DOM_i) = 1 - (1-c_i e_u) \left(\prod_{j=1}^n d_{ij} + b \left(1 - \prod_{j=1}^n d_{ij} \right) \right)$.

□

Theorem 4.12: *The combination of any simple support function about the non determinance of an attribute i with the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} will result in belief and plausibility measures of the DOM_i , $Bel'(DOM_i)$ and $P^*(DOM_i)$ such that:*

$$Bel'(DOM_i) = Bel(DOM_i) \text{ and } P^*(DOM_i) \leq P^*(DOM_i)$$

where $Bel(DOM_i), P^*(DOM_i)$ are the belief and the plausibility of DOM_i estimated from the combination of only the basic probability assignments m_i^{ch}, m_i^a and m_{ij}^{ncs} .

PROOF: According to *theorems 4.7 and 4.11*, we have that:

$$Bel'(DOM_i) = Bel(DOM_i)$$

and

$$Bel'(\overline{DOM}_i) = Bel(\overline{DOM}_i) + (1-e_u c_i) b \left(1 - \prod_{j=1}^n d_{ij} \right)$$

However, $(1-e_u c_i)b(1-\prod_{j=1}^n d_{ij}) \geq 0$

since

$0 \leq c_i \leq 1$ (by *theorem 4.1*)

$0 \leq e_u \leq 1$ (by *theorem 4.2*)

$0 \leq d_{ij} \leq 1$ (by *theorem 4.3*)

$0 \leq b \leq 1$ (by the definition of function s in *theorem 4.11*)

Thus,

$Bel'(\overline{DOM}_i) \geq Bel(\overline{DOM}_i) \rightarrow P'*(DOM_i) \leq P*(DOM_i)$.

□

Theorem 4.13: A bayesian basic probability assignment to the determinance and the non determinance of an attribute i , defined as:

$$s(P) = \begin{cases} b & \text{if } P = \bigcap_{j=1}^n \overline{M}_j \\ 1-b & \text{if } P = \bigcup_{j=1}^n M_j \end{cases}$$

where $0 \leq b \leq 1$ is combinable with the basic probability assignments m_i^{ch}, m_i^a and, m_{ij}^{ncs} only if :

$$(1-b)\prod_{j=1}^n d_{ij} < 1$$

If combinable, their combination results into the following total belief and plausibility of the dominance of i :

$$Bel(DOM_i) = \frac{e_u c_i b + (1-b)(1-\prod_{j=1}^n d_{ij})}{1-(1-b)\prod_{j=1}^n d_{ij}}$$

$$P*(DOM_i) = 1 - \frac{(1-e_u c_i)b}{1-(1-b)\prod_{j=1}^n d_{ij}}$$

where $u = ODC_i$.

PROOF: The core of function s , $Cr(s)$, is the set Θ ($Cr(s) \equiv (\bigcap_{j=1}^n \bar{M}_j) \cup (\bigcup_{j=1}^n M_j) = \Theta$). The intersection of this core with the cores of the basic probability assignments m_i^{ch} , m_i^a , m_{ij}^{ncs} , ($j=1, \dots, n$), which are all equal to Θ , as shown in *theorem 4.5*, also equals Θ . Since Θ is a non empty set, these functions are combinable in any possible order according to *theorem 3.4* in [Sha75], P. 63.

Step 1: First, function s is combined with the functions m_{ij}^{ncs} .

The focals of the m^{ncs} , which have already been derived in *lemma 4.2*, may be distinguished as follows:

$$\bigcap_{j=1}^n \bar{M}_j$$

$$j \in I, I \subset \left\{ \bigcap_{j \in I} \bar{M}_j \mid I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset \right\}, I \neq \emptyset$$

Θ

Also as shown in the same lemma, m^{ncs} , assigns the following basic probabilities to these focals:

$$m^{ncs}(\bigcap_{j=1}^n \bar{M}_j) = \prod_{j=1}^n d_{ij}$$

$$m^{ncs}(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j) = \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) \quad \forall \quad I \subset \{1, 2, \dots, n\}, I \neq \emptyset \text{ and}$$

$$m^{ncs}(\Theta) = \prod_{j=1}^n (1-d_{ij})$$

Thus, the pairwise intersections of the focals of the m^{ncs} with the focals of the function s are:

$$(\bigcap_{j=1}^n \bar{M}_j) \cap (\bigcap_{j=1}^n \bar{M}_j) = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$(\bigcap_{j=1}^n \bar{M}_j) \cap (\bigcup_{j=1}^n M_j) = \emptyset$$

$$(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j) \cap (\bigcap_{j=1}^n \bar{M}_j) = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$\left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j \right) \cap \left(\bigcup_{j=1}^n M_j \right) = \left(\left(\bigcup_{j=1}^n M_j \right) - \left(\bigcap_{j=1}^n \bar{M}_j \right) \right) \neq \emptyset$$

$$\Theta \cap \left(\bigcap_{j=1}^n \bar{M}_j \right) = \bigcap_{j=1}^n \bar{M}_j \neq \emptyset$$

$$\Theta \cap \left(\bigcup_{j=1}^n M_j \right) = \bigcup_{j=1}^n M_j \neq \emptyset$$

Thus, s function can be combined with the function m^{ncs} , according to the rule of the orthogonal sum, only if $k_0 < 1$ (see theorem 3.1 P. 60 in [Sha75]), where k_0 is:

$$k_0 = \sum_{A_p \cap B_q = \emptyset \text{ \& } A_p, B_q \subseteq \Theta} s(A_p) m^{ncs}(B_q) = s\left(\bigcup_{j=1}^n M_j\right) m^{ncs}\left(\bigcap_{j=1}^n \bar{M}_j\right) = (1-b) \prod_{j=1}^n d_{ij} \quad (I)$$

Therefore, we have:

$$m\left(\left(\bigcap_{j=1}^n \bar{M}_j\right) \cap \left(\bigcap_{j=1}^n \bar{M}_j\right)\right) = m^{ncs}\left(\bigcap_{j=1}^n \bar{M}_j\right) s\left(\bigcap_{j=1}^n \bar{M}_j\right) = \frac{\prod_{j=1}^n d_{ij} b}{1-k_0}$$

$$m\left(\left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j\right) \cap \left(\bigcap_{j=1}^n \bar{M}_j\right)\right) = m^{ncs}\left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j\right) s\left(\bigcap_{j=1}^n \bar{M}_j\right) = \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) b}{1-k_0}$$

$$m\left(\left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j\right) \cap \left(\bigcup_{j=1}^n M_j\right)\right) = m^{ncs}\left(\bigcap_{j \in I, I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \bar{M}_j\right) s\left(\bigcup_{j=1}^n M_j\right) = \frac{\prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})(1-b)}{1-k_0}$$

$$m\left(\Theta \cap \left(\bigcap_{j=1}^n \bar{M}_j\right)\right) = m^{ncs}(\Theta) s\left(\bigcap_{j=1}^n \bar{M}_j\right) = \frac{\prod_{j=1}^n (1-d_{ij}) b}{1-k_0}$$

$$m\left(\Theta \cap \left(\bigcup_{j=1}^n M_j\right)\right) = m^{ncs}(\Theta) s\left(\bigcup_{j=1}^n M_j\right) = \frac{\prod_{j=1}^n (1-d_{ij}) b}{1-k_0}$$

Next, these belief estimates are accumulated into belief to DET_i and \overline{DET}_i , according to formula (4.1) as it follows:

$$Bel(DET_i) = \sum_{I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} m\left(\left(\bigcap_{j \in I} \bar{M}_j\right) \cap \left(\bigcup_{j=1}^n M_j\right)\right) + m\left(\Theta \cap \left(\bigcup_{j=1}^n M_j\right)\right) =$$

$$\frac{\sum_{I \subset \{1, 2, \dots, n\} \text{ \& } I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij})(1-b) + \prod_{j=1}^n (1-d_{ij})(1-b)}{1-k_0}$$

However, by lemma 4.1:

$$\sum_{I \subset \{1,2,\dots,n\} \text{ \& } I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) = 1 - \prod_{j=1}^n d_{ij} - \prod_{j=1}^n (1-d_{ij}) \quad (II)$$

Therefore,

$$Bel(DET_i) = \frac{(1-b)(1 - \prod_{j=1}^n d_{ij} - \prod_{j=1}^n (1-d_{ij})) + (1-b) \prod_{j=1}^n (1-d_{ij})}{1-k_0} = \frac{(1-b)(1 - \prod_{j=1}^n d_{ij})}{1-k_0}$$

Also,

$$\begin{aligned} Bel(\overline{DET}_i) &= m((\bigcap_{j=1}^n \overline{M}_j) \cap (\bigcap_{j=1}^n \overline{M}_j)) + \sum_{I \subset \{1,2,\dots,n\} \text{ \& } I \neq \emptyset} m((\bigcap_{j \in I} \overline{M}_j) \cap (\bigcap_{j=1}^n \overline{M}_j)) + m(\Theta \cap (\bigcap_{j=1}^n \overline{M}_j)) = \\ &= \frac{\prod_{j=1}^n d_{ij} b}{1-k_0} + \frac{\sum_{I \subset \{1,2,\dots,n\} \text{ \& } I \neq \emptyset} \prod_{j \in I} d_{ij} \prod_{j \in I^c} (1-d_{ij}) b}{1-k_0} + \frac{\prod_{j=1}^n (1-d_{ij}) b}{1-k_0} \end{aligned}$$

Thus, due to the relation (II):

$$Bel(\overline{DET}_i) = \frac{b}{1-k_0}$$

Step 2: As it shown in theorem 4.7, the combination of the basic probability assignments m_i^{ch} and m_i^a results into the basic probability assignment m' , having the following focals:

$$ABS_i \cap CH_i \quad ABS_i \cap \overline{CH}_i \quad \overline{ABS}_i \cap CH_i \quad \overline{ABS}_i \cap \overline{CH}_i$$

In addition, all the focals of the combination of function s with the m^{ncs} , are subsets of either DET_i or \overline{DET}_i . Therefore, the application of the rule of the orthogonal sum gives (m is the basic probability assignment obtained in step 1):

$$\begin{aligned} m''(ABS_i \cap CH_i \cap DET_i) &= m'(ABS_i \cap CH_i) m(DET_i) = c_i e_u \frac{(1-b)(1 - \prod_{j=1}^n d_{ij})}{(1-k_0)} \\ m''(\overline{ABS}_i \cap CH_i \cap DET_i) &= m'(\overline{ABS}_i \cap CH_i) m(DET_i) = c_i (1-e_u) \frac{(1-b)(1 - \prod_{j=1}^n d_{ij})}{(1-k_0)} \\ m''(ABS_i \cap \overline{CH}_i \cap DET_i) &= m'(ABS_i \cap \overline{CH}_i) m(DET_i) = (1-c_i) e_u \frac{(1-b)(1 - \prod_{j=1}^n d_{ij})}{(1-k_0)} \end{aligned}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i) e_u \frac{(1-b)(1-\prod_{j=1}^n d_{ij})}{(1-k_0)}$$

$$m''(ABS_i \cap CH_i \cap \overline{DET}_i) = m'(ABS_i \cap CH_i) m(\overline{DET}_i) = c_i e_u \frac{b}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap CH_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap CH_i) m(\overline{DET}_i) = c_i (1-e_u) \frac{b}{(1-k_0)}$$

$$m''(ABS_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(ABS_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i) e_u \frac{b}{(1-k_0)}$$

$$m''(\overline{ABS}_i \cap \overline{CH}_i \cap \overline{DET}_i) = m'(\overline{ABS}_i \cap \overline{CH}_i) m(\overline{DET}_i) = (1-c_i)(1-e_u) \frac{b}{(1-k_0)}$$

Then by accumulating these basic probability assignments into total beliefs according to the formula (4.1) we get:

$$\begin{aligned} Bel(DOM_i) &= \sum_{X \subseteq (ABS_i \cap CH_i) \cup \overline{DET}_i} m''(X) = m''((ABS_i \cap CH_i) \cap \overline{DET}_i) + m''((\overline{ABS}_i \cap CH_i) \cap \overline{DET}_i) + \\ & m''((ABS_i \cap \overline{CH}_i) \cap \overline{DET}_i) + m''((\overline{ABS}_i \cap \overline{CH}_i) \cap \overline{DET}_i) + m''((ABS_i \cap CH_i) \cap \overline{DET}_i) = \\ & \frac{(1-b)(1-\prod_{j=1}^n d_{ij})}{(1-k_0)} (c_i e_u + c_i(1-e_u) + (1-c_i) e_u + (1-c_i)(1-e_u)) + c_i e_u \frac{b}{(1-k_0)} = \frac{e_u c_i b + (1-b)(1-\prod_{j=1}^n d_{ij})}{(1-k_0)} \end{aligned}$$

Thus, by substituting k_0 from (I), we have:

$$Bel(DOM_i) = \frac{e_u c_i b + (1-b)(1-\prod_{j=1}^n d_{ij})}{(1-(1-b)\prod_{j=1}^n d_{ij})}$$

Also,

$$\begin{aligned} Bel(\overline{DOM}_i) &= \sum_{X \subseteq (\overline{ABS}_i \cup \overline{CH}_i) \cap \overline{DET}_i} m''(X) = m''((\overline{ABS}_i \cap \overline{CH}_i) \cap \overline{DET}_i) + m''((ABS_i \cap \overline{CH}_i) \cap \overline{DET}_i) + \\ & m''((\overline{ABS}_i \cap CH_i) \cap \overline{DET}_i) = \frac{b}{(1-k_0)} (c_i(1-e_u) + (1-c_i) e_u + (1-c_i)(1-e_u)) = \frac{b(1-e_u c_i)}{(1-k_0)} \end{aligned}$$

Thus, since $P^*(DOM_i) = 1 - Bel(\overline{DOM}_i)$ and by substituting k_0 from (I), we have:

$$P^*(DOM_i) = 1 - \frac{(1-e_u c_i) b}{1-(1-b)\prod_{j=1}^n d_{ij}} .$$

□

Chapter 5

Implementation and Evaluation of the Model

5.1 Introduction

This chapter presents a prototype implementation of the similarity model, an analysis of the computational complexity of the algorithms involved and the results of a preliminary experiment of evaluation.

The similarity model is implemented through a set of algorithms, which compute the distance and the salience measuring functions, introduced in the previous two chapters. These algorithms are specified and analyzed in terms of their time and space complexities. Their implementation constitutes a computational module, which has been integrated with a tool for representing scientific knowledge and engineering designs, the *Semantic Index System*[CD93]. The purpose of this integration, was to provide a prototype for experimentation with tasks of analogical conceptual modeling and design.

This prototype has been used in experiments evaluating:

- the *consistency* of the estimates computed by analysis of similarity with respect to assessments of similarity obtained by humans;

- the *recall* performance of the model as a mechanism of analogical retrieval; and,
- the *time performance* of the model.

The results of these experiments are reported and discussed in this chapter.

5.2 The Algorithms

The analysis of similarity is carried out by three basic algorithms, which compute the distance of two objects with respect to classification (*ClassificationDistance* algorithm), generalization (*GeneralizationDistance* algorithm) and attribution (*AttributionDistance* algorithm). In essence, these algorithms compute the partial distance metric functions d_2 , d_3 and d_4 , defined in the third chapter. The fourth partial distance of the model (i.e. the function d_1) corresponds to a simple identity test over the identifiers of the involved objects.

These partial distances are aggregated into an overall distance measure by a fourth general algorithm (*ObjectDistance* algorithm), which also checks the validity of comparison between two arbitrary objects, according to the principle of *ontological uniformity*.

Figure 5.1 depicts the dependencies between these four algorithms. It also presents their dependencies on other algorithms, which carry out auxiliary computations.

These auxiliary algorithms include:

- i) the *ClassDepth* algorithm, that computes the depth of classes in Isa graphs, which are necessary for the computation of classification and generalization distances ;
- ii) the *GetOriginalClass* algorithm, that retrieves the original classes of attributes, which are necessary for the evaluation of the salience of attributes and their generalization distances ;
- iii) the *GetSalienceSets* algorithm, that retrieves the *scope*, the *refining classes* and the *possible ranges* of attributes, which are necessary for the evaluation of their salience ;
- iv) the *EvaluateSalience* algorithm, that computes the salience of attributes, which is necessary for the computation of attribution distances between objects ;
- v) the *PossibleMappings* algorithm, that checks which of the attributes of two objects could be mapped onto each other and estimates their pairwise distances; and,

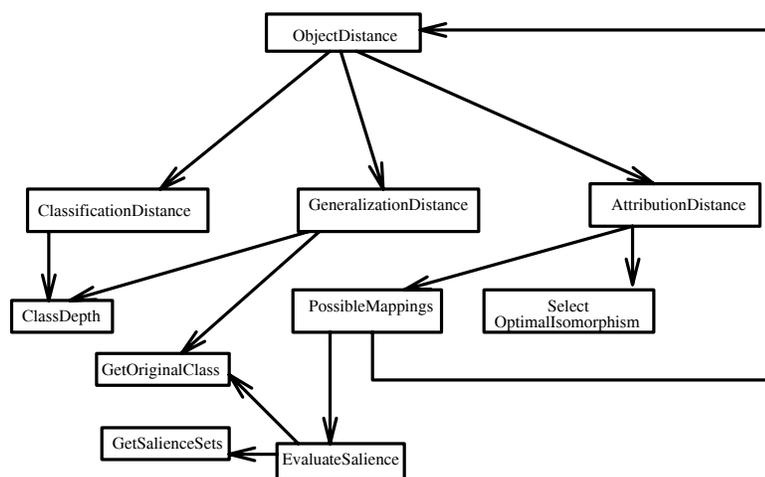


Figure 5.1: Dependencies Between the Algorithms of the Similarity Model

vi) the *SelectOptimalIsomorphism*, that computes the minimum distance isomorphism between the attributes of two objects.

In the following we specify all these algorithms and analyze their worst case time and space complexities. Prior to these specifications we describe two basic structures of sets and objects, which are extensively utilized in them, to let the reader comprehend the subsequent algorithmic specifications. The algorithms of the similarity model are discussed in a bottom-up sequence, in order to let the reader comprehend the gradual analysis of the overall complexity of the model, readily.

5.2.1 Primitive Functions and Operations

As it will be evident in the subsequent sections, the two basic structures utilized in almost all the algorithms for similarity analysis, are a structure for sets and a basic structure for objects which store analogs. These structures and their associated operators and/or functions are described in this section to let the reader understand the subsequent specifications of the algorithms of the model.

Structure of Sets and their Basic Operations. All sets are assumed to be sorted by the value of their elements-identifiers and be structured as doubly linked lists. Primitive set handling is possible through the following operators and functions, at the relevant worst

case time complexities:

- union operator $(S_1 \cup S_2) : O_t(\max\{|S_1|, |S_2|\})$;
- intersection operator $(S_1 \cap S_2) : O_t(\max\{|S_1|, |S_2|\})$;
- difference operator $(S_1 - S_2) : O_t(\max\{|S_1|, |S_2|\})$;
- insert function $(ins(S_1, e)) : O_t(\log|S_1|)$;
- delete function $(del(S_1, e)) : O_t(\log|S_1|)$; and,
- set membership function $(mem(S_1, e)) : O_t(\log|S_1|)$.

The exact details of how these operators and functions are implemented given the prescribed assumption about the structure of sets, and the derivation of the relevant complexities, can be found in[Hal93]. Here, we roughly explain the derivation of the complexity of the union operator, as an example.

Since, sets S_1 and S_2 are structured as sorted lists the formation of their sorted union can start by taking the elements of the set with the smaller first element. Let us assume that this is set S_1 , without loss of generality. Successive elements of S_1 are inserted into the answer set, as long as they are smaller than the first element of S_2 . When the first non smaller element of S_1 is encountered, the insertion proceeds with the elements of S_2 in exactly the same manner, having skipped the last element of S_1 , if it is equal to the first element of S_2 that caused the swapping. Thus, the union of the sets can be formed in at most $|S_1| + |S_2|$ steps and therefore the time complexity of union is $O_t(\max\{|S_1|, |S_2|\})$.

The functions $ins(S,e)$, $del(S,e)$ and $mem(S,e)$ are all based on a binary search, since sets are sorted and thus their complexities are $O_t(\log|S|)$.

Structure of Objects and their Retrieval Functions. Objects in the subsequent algorithms are assumed to be structured as illustrated in figure 5.2. In particular, an object aggregates 6 pointers to sets of identifiers denoting:

- its direct classes(i.e. those classes of the object having no subclass, which is also a class of it);

- its direct instances(i.e. those instances of the object, which are not instances of any of its subclasses);
- its direct superclasses(i.e. those superclasses of the object having no subclass, which is also a superclass of it);
- its direct subclasses(i.e. those subclasses of the object having no superclass, which is also a subclass of it);
- its direct attributes(i.e. those attributes defined or refined in the intension of the object) ; and,
- the attributes having this object as value.

Direct Classes	→ set of identifiers
Direct Instances	→ set of identifiers
Direct Superclasses	→ set of identifiers
Direct Subclasses	→ set of identifiers
Direct AttributesOf	→ set of identifiers
AttributesTo	→ set of identifiers

Figure 5.2: Basic Data Structure of Objects

Given this structure, six primitive querying functions are necessary for retrieving the direct classes, instances, superclasses, subclasses and attributes of objects. These functions are assumed to take as input an identifier denoting an object and return a pointer to a sorted set including the identifiers of the objects that constitute the answer. The six primitive querying functions are:

get_classes(s): It retrieves the direct classes of an object with identifier *s* in a single step.

get_instances(s): It retrieves the direct instances of an object with identifier *s* in a single step.

get_superclasses(s): It retrieves the direct superclasses of an object with identifier *s* in a single step.

get_subclasses(s): It retrieves the direct subclasses of an object with identifier *s* in a single step.

get_attributes_of(s): It retrieves the attributes of an object with identifier *s* in a single step.

get_attributes_to(s): It retrieves the attributes having as value an object with identifier *s* in a single step.

These primitive querying functions, are used by other *transitive querying functions*, which retrieve the transitive closures of the classes, the instances, the superclasses and the subclasses of an object.

All these functions, operate in a breadth-first manner, as specified by the abstract *TransitiveClosure* retrieval algorithm, illustrated in figure 5.3.

ALGORITHM: TransitiveClosure(*#s*,TC)
INPUT: *#s* /* the identifier of an object */
OUTPUT: TC /* a set containing the identifiers of objects in the relevant transitive closure */
STEPS:
TC = \emptyset ;
S = primitive_get_function1(*#s*); /* S is the set with the identifiers of the objects at the current depth of the graph being traversed */
while S $\neq \emptyset$ **do**
 TC = TC \cup S ;
 NEXT = \emptyset ;
 for each *#i* in S **do**
 NEXT = NEXT \cup primitive_get_function2(*#i*) ;
 endfor
 S = NEXT;
endwhile

Figure 5.3: Abstract TransitiveClosure Retrieval Algorithm

The worst time complexity of the abstract *TransitiveClosure* retrieval algorithm is $O_i(N^2)$, where N is the number of the nodes in the relevant graph. This complexity is derived as

follows.

In the inner iteration, an object identified by #i may have at most N-1 objects directly connected to itself by the relations being traversed, since N is the number of all the nodes in the transitive closure of #s. So N is the maximum size of the set retrieved by the *primitive_get_function2*. Similarly, the set NEXT may have at most this size. Thus, the computation of their union will take at most N steps(see the complexity of the union operator above).

Also, the very assumption about the structuring of objects implies that each node in their classification and generalization/specialization graphs is connected to themselves(i.e. the roots of the graphs) with paths of exactly the same length. This is because each object stores the identifiers of only its direct classes, instances, superclasses and subclasses. According to these lengths nodes can be partitioned in successive levels. These levels are visited only once, during the breadth-first traversal. Thus, regardless of the number of nodes at successive levels, the total number of nodes of all levels will be N. Consequently, the outer iteration of the algorithm will be performed at most N times.

The worst space complexity of the algorithm is $O_s(N)$, since N is the maximum size of the three sets that must be maintained by the algorithm(sets TC, S and NEXT).

On the basis of the abstract *TransitiveClosure* retrieval algorithm we can define the following four *transitive querying functions*:

- *get_all_classes(#s)*: This function retrieves all the classes of an object identified by #s. It is defined by substituting the *primitive_get_function1* and the *primitive_get_function2* in the *TransitiveClosure* retrieval algorithm, by the primitive querying functions *get_classes* and *get_superclasses*, respectively. Its worst case time and space complexities are $O_t(N^2)$ and $O_s(N)$, respectively, where N is the number of all the classes of #s ;
- *get_all_superclasses(#s)*: This function retrieves all the superclasses of an object identified by #s. It is defined by substituting both the *primitive_get_function1* and the *primitive_get_function2* in the *TransitiveClosure* retrieval algorithm, by the primitive querying function *get_superclasses*. Its worst case time and space complexities are $O_t(N^2)$ and $O_s(N)$, respectively, where N is the number of all the superclasses of #s ;

- *get_all_subclasses(#s)*: This function retrieves all the subclasses of an object identified by #s. It is defined by substituting both the *primitive_get_function1* and the *primitive_get_function2* in the *TransitiveClosure* retrieval algorithm, by the primitive querying function *get_subclasses*. Its worst case time and space complexities are $O_t(N^2)$ and $O_s(N)$, respectively, where N is the number of all the subclasses of #s ; and,
- *get_all_instances(#s)*: This function retrieves the transitive closure of the instances of an object identified by #s, as specified in figure 5.4. The worst time complexity of the algorithm is $O_t(N^2 + NN_i)$, where N is the number of all the subclasses of #s and N_i is the number of all its instances. This complexity is derived in exactly the same way with the complexity of the *TransitiveClosure* retrieval algorithm, noticing that the inner iteration in figure 5.4, has one more union operator, whose complexity is N_i (i.e. the $TC=TC \cup get_instances(\#i)$). This is because the involved sets may have at most N_i elements. The worst space complexity of the function is $O_s(\max\{N_i, 2N\})$, since the set TC may have N_i elements at maximum and, the sets S, NEXT may have N elements at maximum.

ALGORITHM: *get_all_instances(#s,TC)*
INPUT: #s /* the identifier of a class */
OUTPUT: TC /* a set containing the identifiers of all its instances*/
STEPS:
TC = *get_instances(#s)*;
S = *get_subclasses(#s)*; /* S is the set with the identifiers of the subclasses at the current depth of the specialization graph of #s */
while S $\neq \emptyset$ **do**
 NEXT = \emptyset ;
 for each #i in S **do**
 NEXT = NEXT \cup *get_subclasses(#i)* ;
 TC = TC \cup *get_instances(#i)* ;
 endfor
 S = NEXT;
endwhile

Figure 5.4: *get_all_instances* Algorithm

5.2.2 The ClassDepth Algorithm

The depth of a class is required for the evaluation of both the classification and generalization distances between objects. This depth is estimated as the maximum length of the generalization paths connecting a class with its *Instantiation Level Superclass* (see

definition 3.22 in chapter 3). The *ClassDepth* algorithm computes this maximum length by computing the lengths of all the existing paths from a class to its instantiation level superclass, in a breadth-first sequence, as specified in figure 5.5.

```

ALGORITHM: ClassDepth(#s,D)
INPUT: #s /* the identifier of a class */
OUTPUT: D /* the depth of #s */
STEPS:
S = get_superclasses(#s); D = 1 ;
while S ≠ ∅ do
  D = D + 1 ;
  PS = S ; S = ∅ ;
  for each #c in PS do
    S = S ∪ get_superclasses(#c) ;
  endfor
endwhile

```

Figure 5.5: ClassDepth Algorithm

Since the algorithm traverses in a breadth-first sequence, the transitive closure of the superclasses of #s, its worst case time complexity is $O_t(N^2)$, where N is the number of all the superclasses of #s, as discussed in section 5.2.1 . Similarly the worst case space complexity of the algorithm is $O_s(N)$.

5.2.3 The GetOriginalClass Algorithm

The *GetOriginalClass* algorithm scans in a breadth-first sequence all the superclasses of an attribute class, retaining only those which have identical logical names with it. It terminates upon the presence of no more such superclasses and returns the last retrieved one as the original class of the input attribute class. Notice that, due to axiom A.3.18 introduced in chapter 3, it is guaranteed that there will never exist more than one original attribute classes, for any input attribute class.

The algorithm is specified in figure 5.6 .

The worst case time complexity of the *GetOriginalClass* algorithm is $O_t(N \log N)$, where N is the number of all the superclasses of the attribute #s. This is because the algorithm traverses these superclasses in a breath-first sequence(see section 5.2.1) and inserts those satisfying the criterion of the identical labels, into the set NEXT whose size is bounded by N. As discussed in section 5.2.1, the insertion in a set of size N, takes at most $\log N$

ALGORITHM: GetOriginalClass(#s,#c)
INPUT: #s /* an identifier of an attribute class */
OUTPUT: #c /* the identifier of the original class of #s */
STEPS:
 $C = \emptyset$; ins(C,#s); crd = 3;
while (crd > 2) **do**
 NEXT = \emptyset ;
 for each #i in C **do**
 SC = get_superclasses(#i);
 for each #j in SC **do**
 if label(#j)=label(#s) **then** ins(NEXT,#j) ; **endif**
 /* label(#i) returns the logical name of an attribute identified by #i */
 endfor
 endfor
 crd = | NEXT | ; C = NEXT ;
endwhile
#c = get_first_element(C); /* C is guaranteed to have a single element by axiom A.3.18 in chapter 3 */

Figure 5.6: GetOriginalClass Algorithm

steps. Also, the worst space complexity of the *GetOriginalClass* algorithm, as a breadth first transitive closure traversing algorithm is $O_s(N)$.

5.2.4 The ClassificationDistance Algorithm

The algorithm computing the classification distance of two objects initially retrieves the transitive closures of the classes of the objects, computes their symmetric difference and takes the sum of the inverse depths of the classes in this difference.

Its specification is presented in figure 5.7.

The worst case time complexity of the *ClassificationDistance* algorithm is $O_t(N_1^2 + N_2^2 + 2\max\{N_1, N_2\} + N_1^3 + N_2^3)$. The factors N_1^2, N_2^2 express the cost of retrieving all the N_1 classes of #s1 and the N_2 classes of #s2 (see section 5.2.1). The factor $2\max\{N_1, N_2\}$ is the cost of evaluating the symmetric difference of these closures (i.e. computation of sets S12 and S21 in figure 5.7), as indicated in section 5.2.1. Finally, the factors N_1^3, N_2^3 express the costs of finding the depths of the classes in sets S12 and S21. Notice that S12 and S21 may have at most N_1 and N_2 elements (if #s1 and #s2 share no common classes). Thus, the algorithm *ClassDepth* will be invoked at most N_1 and N_2 times. Furthermore, as discussed in section 5.2.2, the cost of evaluating the depths of each class in set S12 is

ALGORITHM: ClassificationDistance($\beta_2, \#s1, \#s2, d$)

INPUT: β_2 /* a real number greater than 0 used for the homographic transformation of absolute classification distances into relative ones */

$\#s1, \#s2$ /* the unique identifiers of the objects under comparison */

OUTPUT: d /* the relative classification distance of the objects identified by $\#s1$ and $\#s2$ */

STEPS:

```

d = 0;
S1 = get_all_classes(#s1);
S2 = get_all_classes(#s2);
S12 = S1 - S2;
S21 = S2 - S1;
for each #e in S12 do
  ClassDepth(#e,D);
  d = d + 1/D;
endfor
for each #e in S21 do
  ClassDepth(#e,D);
  d = d + 1/D;
endfor
d =  $\beta_2 d / (\beta_2 d + 1)$ 

```

Figure 5.7: ClassificationDistance Algorithm

$O_t(N_1^2)$ since each of the classes in this set has at most N_1 superclasses. Similarly, the cost of evaluating the depth of each class in S21 will be $O_t(N_2^2)$.

The worst case space complexity of the classification distance algorithm is $O_s(N_1 + N_2)$ due to the maintenance of sets S1, S12 and S2, S21.

5.2.5 The GeneralizationDistance Algorithm

The algorithm computing the generalization distance between objects first checks whether the input objects are both individuals or attributes. In the case of individuals, it retrieves the transitive closures of their superclasses, computes their symmetric difference and estimates the sums of the inverse depths of the classes in this difference. In the case of attributes, it retrieves their original classes and sets their generalization distance equal to 0, if these original classes are identical or equal to 1, if they are not.

The algorithm is specified in figure 5.8.

The worst case time and space complexities of the generalization distance algorithm, are derived as in the case of the classification distance algorithm, when the objects under comparison are individuals. Thus, the time complexity is $O_t(N_1^2 + N_2^2 + 2 \max\{N_1, N_2\} + N_1^3 + N_2^3)$

ALGORITHM: GeneralizationDistance($\beta_3, \#s1, \#s2, d$)

INPUT: β_3 /* a real number greater than 0 used for the homographic transformation of absolute generalization distances into relative ones */

$\#s1, \#s2$ /* the unique identifiers of the objects under comparison */

OUTPUT: d /* the relative generalization distance of the objects identified by $\#s1$ and $\#s2$ */

STEPS:

```

d = 0;
if  $\#s1$  and  $\#s2$  are attribute classes then
  GetOriginalClass( $\#s1, \#c1$ ); GetOriginalClass( $\#s2, \#c2$ );
  if  $\#c1 \neq \#c2$  then
    d = 1;
  endif
else
  S1 = get_all_superclasses( $\#s1$ );
  S2 = get_all_superclasses( $\#s2$ );
  S12 = S1 - S2;
  S21 = S2 - S1;
  for each  $\#e$  in S12 do
    ClassDepth( $\#e, D$ );
    d = d + 1/D;
  endfor
  for each  $\#e$  in S21 do
    ClassDepth( $\#e, D$ );
    d = d + 1/D;
  endfor
  d =  $\beta_3 d / (\beta_3 d + 1)$ 
endif

```

Figure 5.8: GeneralizationDistance Algorithm

and the space complexity is $O_s(N_1 + N_2)$. In this case, N_1 and N_2 are the numbers of the superclasses of the involved objects.

In the case of attribute objects, the worst case time and space complexities are $O_t(N_1 \log N_1 + N_2 \log N_2)$ and $O_s(N_1 + N_2)$. These complexities result from the retrieval of the original classes of the involved attributes, as discussed in section 5.2.3. The worst case complexity of this algorithm will be referred to as K_{isa} , henceforth.

5.2.6 The GetSalienceSets Algorithm

The retrieval of the scope, the refining classes and the possible ranges of an attribute is carried out by the *GetSalienceSets* algorithm. Initially, the algorithm accumulates all the subclasses of the domain class of the input attribute class, into the scope of this attribute class (i.e. the set S). Then, it retrieves in a breadth-first sequence, the subclasses of the input attribute class itself and inserts their range-classes into the set of the possible

ranges(i.e. the set AR).

For each of these attribute subclasses, it also tests whether it has an identical logical name with the input attribute class and, if it does, the domain class of the attribute is inserted into the set of the refining classes(i.e. the set R).

The *GetSaliencSets* algorithm is specified in figure 5.9

ALGORITHM: GetSaliencSets(#s,S,R,AR)

INPUT: #s /* the identifier of an attribute class , which is the original class of the attribute whose salienc sets will be retrieved */

OUTPUT: S /* a set with the classes in the scope of #s */

R /* a set with the refining classes of #s */

AR /* a set with the possible class ranges of #s */

STEPS:

$$S = \text{get_all_subclasses}(o(\#s).FROM) ; R = \{o(\#s).FROM\} ; AR = \{o(\#s).TO\} ; CLS = \{\#s\} ;$$

while CLS $\neq \emptyset$ **do**

 NEXTSUB = \emptyset ;

for each #l in CLS **do**

 SUB = get_subclasses(#l);

for each #sl in SUB **do**

if label(#sl)=label(#s) **then**

 ins(NEXTSUB,#sl);

 ins(R,o(#sl).FROM);

 ins(AR,o(#sl).TO) ;

endif

endfor

endfor

 CLS = NEXTSUB ;

endwhile

Figure 5.9: GetSaliencSets Algorithm

The worst case time complexity of the *GetSaliencSets* algorithm is $O_i(N_1^2 + N_2^2 \log N_2)$. N_1 is the number of all the subclasses of the domain class of the input attribute class(i.e. the class $o(\#s).FROM$). N_2 is the number of all the subclasses of #s. The factor N_1^2 is the cost of retrieving the transitive closure of the subclasses of the domain class of #s, as discussed in section 5.2.1. The factor $N_2^2 \log N_2$ is the cost of traversing the transitive closure of the subclasses of #s(i.e. the outermost iteration in figure 5.9). As discussed in section 5.2.1 this traversal takes at most N_2^2 steps. Also, at each such step the insertion of #sl into the set NEXTSUB will take at most $\log N_2$ steps, since NEXTSUB has at most N_2 elements(see *ins(S,e)* function in section 5.2.1).

The worst case space complexity of the algorithm is $O_s(N_1+N_2)$. The space required for storing the sets S, R and AR is bounded by $3N_1$. The size of the first of these sets is equal to N_1 , while the sizes of the latter two may also be equal to this number. Also a space of size $3N_2$ is the space needed for storing subclasses of the input attribute class of two successive levels(i.e. sets CLS and SUB) and for accumulating the identically labeled subclasses into the set NEXTSUB. Each of these sets has a size bounded by N_2 .

5.2.7 The EvaluateSaliience Algorithm

The algorithm *EvaluateSaliience* computes the saliience of attribute classes as the mean of their range of belief about their dominance(see section 4.4.2 in chapter 4).

Initially, the algorithm retrieves the original class of an attribute given as input to it and the three sets of classes, which are used in the evaluation of the belief range of an attribute's dominance (i.e. the *scope*, the *refining classes* and the *possible ranges* of the attribute). The beliefs to the properties of *charactericity* and *abstractness* of an attribute are estimated according to the relevant functions defined in the fourth chapter.

Next, the algorithm retrieves the *direct extension*(i.e. the instances of a class, which are not instances of any of its subclasses) and the *extension*(i.e. the set of all the instances of a class) of the original domain class of the input attribute. These two sets are used in the estimation of the belief to its abstractness according to definition 4.13 in chapter 4.

Finally, the algorithm retrieves the original classes of all the other attributes of the classes in the scope of the input attribute, and the scope, the refining and the possible range classes of these original classes. These sets are used in estimating partial beliefs to the *non determinance* of the input attribute. Eventually, partial beliefs are aggregated into a global belief to non determinance of the input attribute. This belief is combined with the beliefs to the charactericity and abstractness of the attribute resulting into the boundaries of its saliience range(see theorem 4.7 in chapter 4).

The *EvaluateSaliience* algorithm is specified in figure 5.10.

The worst case time complexity of the algorithm is $O_t((N_1 \log N_1) + (N_2^2 + N_3^2 \log N_3) + (N_4 N_2 + N_2^2) + N_4 + (\sum_a N_a \log N_a) + (\sum_{a'} N_{2a'}^2 + N_{3a'}^2 \log N_{3a'}) + (\sum_{a'} N_2 \log N_{2a'}))$

ALGORITHM: EvaluateSaliency(#s,sal)
INPUT: #s /* a unique identifier of an attribute class */
OUTPUT: sal /* a real value reflecting the saliency of #s */
STEPS:
 GetOriginalClass(#s,#oc); GetSaliencySets(#s,S,R,AR);
 $B_{ch} = |AR| / |S|$;
 $EXT_d = \text{get_instances}(o(\#oc).FROM)$; $EXT = \text{get_all_instances}(o(\#oc).FROM)$;
 $EXT_s = EXT - EXT_d$;
 $B_{abs} = |EXT_s| / |EXT|$;
 $B_{dom} = B_{ch} B_{abs}$;
 $P^*_{dom} = 1 - B_{dom}$;
for each #c in S do
 A = get_attributes(#c);
for each #a in A do
 GetOriginalClass(#a,#a'); ins(A',#a');
endfor
endfor
for each #a' in A' do
 $S' = \emptyset$; $R' = \emptyset$; $AR' = \emptyset$; $crd1 = 0$;
 GetSaliencySets(#a',S',R',AR'); $crd3 = |S'|$;
for each #c in S do
if not(mem(S',#c)) then $crd1 = crd1 + 1$; **else** del(S',#c); **endif**
endfor
 $crd2 = |S'|$; $d_{a'} = (crd1 + crd2) / (crd1 + crd3)$;
 $P^*_{dom} = P^*_{dom} d_{a'}$;
endfor
 $sal = (B_{dom} + P^*_{dom}) / 2$;

Figure 5.10: EvaluateSaliency Algorithm

where:

- N_1 is the number of all the superclasses of attribute #s
- N_2 is the number of the classes in the scope of attribute #s
- N_3 is the number of all the subclasses of the original class of attribute #s
- N_4 is the number of all the instances in the extension of the original domain class of attribute #s
- N_a is the number of the superclasses of each attribute #a associated with some class in the scope of attribute #s
- $N_{2a'}$ is the number of the classes in the scope of attribute #a'
- $N_{3a'}$ is the number of all the subclasses of the original class of attribute #a'

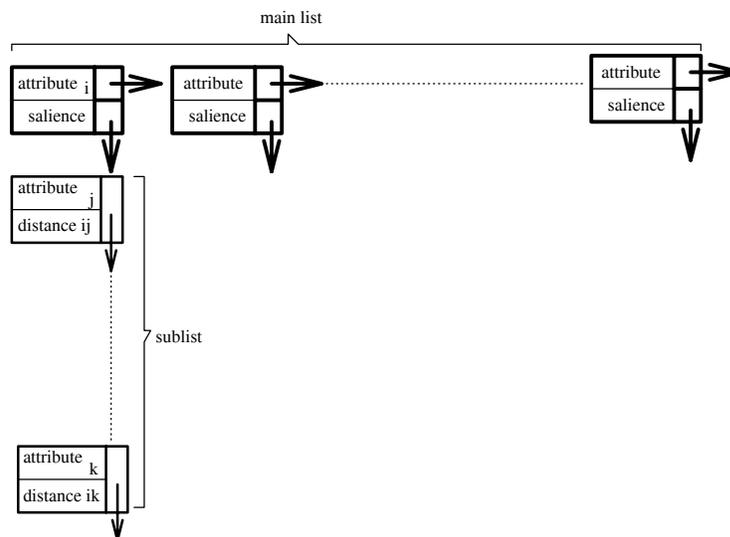
The previous complexity measure aggregates the following cost factors:

- i) $(N_1 \log N_1)$ is the cost of retrieving the original class of the attribute #s
- ii) $(N_2^2 + N_3^2 \log N_3)$ is the cost of retrieving the salience sets S,R and AR of the attribute #s
- iii) $(N_4 N_2 + N_2^2)$ is the cost of retrieving all the instances of the domain class of the original class of #s (set `get_all_instances` function in section 5.2.1)
- iv) N_4 is the cost of evaluating the difference of the sets EXT and EXT_d (see set difference operator in section 5.2.1)
- v) $(\sum_a N_a \log N_a)$ is the cost of retrieving the original classes a' of all the attributes #a of the classes in the scope of #s
- vi) $(\sum_{a'} N_{2a'}^2 + N_{3a'}^2 \log N_{3a'})$ is the cost of retrieving the salience sets S',R' and AR' of all #a'
- vii) $(\sum_{a'} N_2 \log N_{2a'})$ is the cost of testing whether the classes in the scope of #s also belong to the scopes of all #a'

The worst case space complexity of the *EvaluateSalience* algorithm is $O_s(2N_2 + \max\{2N_4, N_{A'} + \max\{N_{2a'}\}\})$. This space complexity results from the need to maintain all the sets S, and AR (whose sizes may all be at most N_2) together with either the sets EXT and EXT_d (whose sizes may be at most N_4) or the set A' of all the original classes of the attributes of the classes in the scope of #s (whose size is equal to $N_{A'}$) and then the sets S' of all the attributes in A' (whose size is $N_{2a'}$).

5.2.8 The PossibleMappings Algorithm

The *PossibleMappings* algorithm identifies pairs of attributes of two objects A and B, that could be mapped onto each other. It proceeds by examining the attributes of object A. For each of these attributes it estimates its salience, identifies the attributes of B that could be mapped on and computes its pairwise distances with these attributes. All this information is stored in a structure presented in figure 5.11. The nodes in the main list of this structure contain the identifiers and salience estimates of the attributes of A and point to a secondary list, whose nodes contain the attributes of B, they can map on, and their pairwise distances.



functions:

`get_sublist(MLP*,#i,L*)`: it returns the sublist of the node of the main list with attribute value `#i`

`delete_sublist(MLP*,#i)`: it deletes the sublist of the node of the main list with attribute value `#i`

`insert_sublist(MLP*,#i,#j,d)`: it inserts a node in the sublist of `#i` with value `(#j,d)`

`insert_main_list(MLP*,#i,s)`: it inserts a node in main list with value `(#i,s)`

`length(MPL*)`: it returns the length of the main list `MLP*`

`get_saliency(MLP*,#i,s)`: it returns the saliency `s` of the main list node with attribute value `#i`

Figure 5.11: A Multilist Structure Used in Specifications of Algorithms

The potential counterparts of an attribute result from the criterion of semantic homogeneity. According to this criterion, the algorithm first retrieves the original class for each of the classes of an attribute. It also retrieves the original classes of the classes of any other attribute which is a candidate for mapping. Only if the two sets of original classes are equal the semantic homogeneity criterion is satisfied and the attributes can be mapped onto each other. Otherwise they cannot.

In cases where an attribute `j` of `B`, which satisfies the semantic homogeneity criterion regarding an attribute `i` of `A`, also has the same original class with `i`, `j` is maintained as the only potential counterpart for `i` (i.e. the algorithm does not maintain the other potential mappings of `i`, which had been identified prior to `j`, neither searches for further potential mappings of `i`). This is due to theorem 3.10 in chapter 3, which guarantees that the optimal isomorphism between the attributes of `A` and `B`, will necessarily include the

ALGORITHM: PossibleMappings($V_s, S1, S2, PM$)

INPUT: V_s /* a set used to store identifiers of objects currently being under comparison, so as to preclude the comparison of attributes having these objects as values(i.e. cyclic attributes), see definition 3.34 in chapter 3 */

$S1$ /* a set with identifiers denoting the attributes of one of the objects under comparison */

$S2$ /* a set with identifiers denoting the attributes of the other object under comparison */

OUTPUT: PM /* a list of including the salience estimates for elements of $S1$, their potential images in $S2$ and their pairwise distances, structured as illustrated in figure 5.11 */

STEPS:

for each #i in $S1$ **do**

$C1i = \text{get_all_classes}(\#i)$; $\text{max_sal}=0$;

for each #c1 in $C1i$ **do**

$\text{GetOriginalClass}(\#c1, \#oc1)$; $\text{EvaluateSalience}(\#oc1, \text{sal})$;

 /* evaluate the salience of the attribute as the max salience of its classes */

if $\text{max_sal} < \text{sal}$ **then** $\text{max_sal} = \text{sal}$; **endif**

$\text{ins}(C1i', \#oc1)$;

endfor

$\text{insert_in_main_list}(PM, \#i, \text{max_sal})$;

endfor

for each #j in $S2$ **do** /* get original classes of attributes in $S2$ */

$C2j = \text{get_all_classes}(\#j)$;

 /* identify the original classes of the classes of #j */

for each #c2 in $C2j$ **do**

$\text{GetOriginalClass}(\#c2, \#oc2)$; $\text{ins}(C2j', \#oc2)$;

endfor

endfor

/* check the criterion of semantic homogeneity */

for each #i in $S1$ **do**

for each #j in $S2$ **do**

 /* check the criterion of the semantic homogeneity */

if $C1i' = C2j'$ **then**

$\text{ObjectDistance}(V_s, \#i, \#j, I, d)$;

$\text{GetOriginalClass}(\#i, \#oc3)$; $\text{GetOriginalClass}(\#j, \#oc4)$;

 /* check the criterion of the common original class */

if $\#oc3 = \#oc4$ **then**

$\text{delete_sublist}(PM, \#i)$; /* delete the so far identified potential maps of #i */

$\text{insert_sublist}(PM, \#i, \#j, d)$; /* maintain only the map of #i to #j */

exit for ; /* do not search for further potential maps */

else

$\text{insert_sublist}(PM, \#i, \#j, d)$;

endif

endif

endfor

endfor

Figure 5.12: PossibleMappings Algorithm

mappings between pairs of semantically homogeneous attributes, which share a common original class. Thus, the other potential mappings of these attributes are not maintained so as to reduce the search space for the selection of this optimal isomorphism(see

algorithm *SelectOptimalIsomorphism* below).

The *PossibleMappings* algorithm is specified in figure 5.12.

The worst case time complexity of the algorithm is $O_t(\sum_i x_i \sum_i y_i K_{map} + \sum_i x_i K_{sal} + \sum_i x_i y_i K_D)$.

The sums $\sum_i x_i$ and $\sum_i y_i$ are the numbers of the attributes in sets S1 and S2 (i.e. the numbers of attributes of the objects under comparison). The subscript i denotes the disjoint categories of the semantically homogeneous attributes of the involved objects.

K_{map} is the cost of testing the criterion of the semantic homogeneity, given by the following formula:

$$K_{map} = \max_{x \in S1} \left\{ \max_{x \in S2} \left\{ N_x^2 + N_x \log N_x + \sum_{z \in \text{Classes}(x)} N_z \log N_z \right\} + \left\{ N_y^2 + N_y \log N_y + \sum_{w \in \text{Classes}(y)} N_w \log N_w \right\} + \max \left\{ N'_x, N'_y \right\} \right\}$$

where

- $N_x(N_y, N_z, N_w)$ is the number of all the classes of attribute x(y,z,w); and,
- $N'_x(N'_y)$ is the number of the original classes of all the classes of attribute x(y) ($N'_x \leq N_x$).

The factor $\left\{ N_x^2 + N_x \log N_x + \sum_{z \in \text{Classes}(x)} N_z \log N_z \right\}$ expresses the cost of:

- retrieving the classes of attribute x in S1 (N_x^2);
- retrieving the original classes of attribute x in S1 ($N_x \log N_x$); and,
- retrieving the original classes of the classes z of attribute x in S1 ($\sum_{z \in \text{Classes}(x)} N_z \log N_z$).

Similarly, the factor $\left\{ N_y^2 + N_y \log N_y + \sum_{w \in \text{Classes}(y)} N_w \log N_w \right\}$ expresses the cost of the same computations for attribute y in S2.

The factor $\max \left\{ N'_x, N'_y \right\}$ expresses the cost of testing the equality of the original classes of the pair of attributes (x,y) from the sets S1 and S2.

The factor K_{sal} is the maximum of the costs of estimating the salience of attributes in S1, given by the formula

$$K_{sal} = \max_{x \in S1} \left\{ N'_x K_{sal}^{(x)} \right\}$$

where $K_{sal}^{(x)}$ is the worst case complexity of the algorithm *EvaluateSaliency* for each attribute x (see section 5.2.7).

Finally, the factor K_D is the maximum of the costs of estimating the pairwise distances between the attributes in S1 and S2, given by the formula

$$K_D = \max_{x \in S1, y \in S2} \left\{ K_D^{(x,y)} \right\}$$

where $K_D^{(x,y)}$ is the computational cost of the *ObjectDistance* algorithm(see formula 5.5 in section 5.2.11 below).

The worst case time complexity of the algorithm is bounded by $O_l(N_1N_2(K_{map} + K_D) + N_1K_{sal})$, where N_1 and N_2 are the numbers of the attributes of the involved objects(i.e. $N_1 = \sum_i x_i$ and $N_2 = \sum_i y_i$). This is because $\sum_i x_i y_i < \sum_i x_i \sum_i y_i$.

The worst case space complexity of the algorithm is $O_s(\max\left\{ \sum_{x \in S1} N'_x + \sum_{y \in S2} N'_y, N_1N_2 \right\})$. This space is required since it is necessary to keep the sets with the original classes of the classes of the attributes in both sets S1 and S2 ($\sum_{x \in S1} N'_x + \sum_{y \in S2} N'_y$) initially, and then, all the pairwise distances between the attributes that can be mapped onto each other(i.e. the factor N_1N_2).

5.2.9 The SelectOptimalIsomorphism Algorithm

The *SelectOptimalIsomorphism* algorithm is a *branch-and-bound* algorithm[PS82], which enumerates the possible isomorphisms between the semantically homogeneous attributes of two objects, and selects the one with the smallest overall distance.

The possible isomorphisms between the attributes of two objects A and B, may be structured as a tree. Each node n in the tree is a pair (i,j) presenting the mapping of an attribute i of A onto an attribute j of B or onto no attribute of B(i.e. the *nil* mapping, $j=nil$). The siblings of a node n express alternative mappings of i onto other attributes of B. The immediate successor and ancestor nodes of node n present possible mappings of the next and the previous attributes of A(attributes can be ordered in an arbitrary linear order). No path of this tree can have more than one node-pairs with the same j argument. Then, each path from the root to some leaf of the tree presents a different isomorphism between the

attributes of A and B.

The algorithm proceeds in a breadth-first enumeration of the paths of the tree (i.e. branching). By convention, the object having the largest number of attributes is selected in the place of object A. When generating the children of a node of an arbitrary path, the algorithm selects the next attribute of object A, and computes the difference between the set of all the attributes of B which this attribute could be mapped on (this set has been computed by the *PossibleMappings* algorithm) and the attributes of B that have been used as counterparts of the previous attributes of A, in the path. For each of these attributes and the *nil* attribute it creates a new child node. The mapping to *nil* attribute is considered so as to ensure that, the yet not searched attributes of A could be mapped onto attributes of B, semantically homogeneous to themselves, which have been already utilized as images of searched attributes of A, during the subsequent enumerations.

The algorithm exploits a *bounding* condition, for pruning paths which can never lead to an optimal isomorphism. According to this condition, if the sum of the distances between the pairs of attributes in a path and the distance between the pair of attributes that is to be added (this sum is called *current overall distance* of the path), exceeds an overall maximum upper bound estimate for the distance of the optimal isomorphism (i.e. the *bounding estimate*), the particular child node is created but the the path including this new node is marked as killed. Killed paths are not taken into account in subsequent enumerations.

The bounding estimate may be updated each time a new child node is added to some path of the tree. This update is subject to a test. If the current overall distance of a path plus the number of the attributes of A, which have not been mapped in this path yet (i.e. the *current overall distance bound* of the path), is lower than the current bounding estimate, this estimate is set equal to the current overall distance bound of the path. This gives a conservative, safe estimate for the upper bound of the distance of the optimal isomorphism. In fact, the distances between the so far not mapped attributes of A in a path, and any of the attributes of B, cannot be greater than 1 (see definition 3.34 in chapter 3). Thus, their approximation by 1 is safe.

Along with the enumeration of paths, the algorithm also keeps a pointer to the path with the minimum current overall distance bound. At the end of search, this bound will coincide with the overall distance of the path because there will be no more attributes of A to

ALGORITHM: SelectOptimalIsomorphism(PM,I,d)

INPUT: PM /* a list including the attributes of an object A, their salience estimates, their potential images in the set of attributes of another object B and their pairwise distances, structured as illustrated in figure 5.11 */

OUTPUT: I /* the optimal isomorphism between the attributes of S1 and the attributes of the object S2; it is a list of 3-tuples (#u,#v,d_{uv}) where #u,#v are mapped attributes and d_{uv} is their distance */

d /* the overall distance of I */

STEPS:

T = empty tree; M = length(PM);

minUD = ∞; /* minUD is the bounding estimate */

for each attribute #i in list PM **do**

 get_salience(PM,#i,sal); get_sublist(PM,#i,L);

 /* L is a list of pairs (#j,d_{ij}) where #j is an attribute #i can be mapped on and d_{ij} their distance */

if T is not empty **then**

 /* T is a tree of 3-tuples (#u,#v,d) where #u,#v are mapped attributes and d is their distance */

for each non killed path t in T **do** /* t is killed if its leaf node is marked as killed */

 U = set of elements #v in the tuples of the path t;

 cld = $\sum_{(\#u,\#v,d) \in t} d$; /* cld is the current overall distance of t */

 /* update path t by enumerating possible mappings of #i */

for each element (#j,d_{ij}) in list L **do**

if (not(mem(U,#j))) **then**

 append a new child node (#i,#j,d_{ij}) to t;

 ncUb = M - length(t) - 1 + (cld + d_{ij}); /* ncUb is current overall distance bound of t */

if ncUb ≤ minUD **then** minUD = ncUb; I = t; **endif** /* test and update the bounding estimate */

if (cld + d_{ij}) > minUD **then**

 mark new child node (#i,#j,d_{ij}) of t as killed;

endif

endif

endfor

 append a new child node (#i,nil, sal²) to t;

if ((cld + sal²) ≤ minUD) **then**

 ncUb = M - length(t) - 1 + cld + sal²;

if ncUb ≤ minUD **then** minUD = ncUb; I = t; **endif**

else

 mark new child node (#i,nil, sal²) of t as killed;

endif

endfor

else /* no paths have been created so far */

for each element (#j,d_{ij}) in list L **do**

 create a new path t in T with initial node (#i,#j,d_{ij});

 cUb = M - 1 + d_{ij}; /* cUb is the current overall distance bound of t */

if cUb ≤ minUD **then** minUD = cUb; I = t; **endif**

endfor

 create a new combination t in T with initial node (#i,nil,sal²);

endif

endfor

d = minUB;

Figure 5.13: SelectOptimalIsomorphism Algorithm

be mapped. Thus, the isomorphism represented by it will be the optimal one.

The *SelectOptimalIsomorphism* algorithm is specified in figure 5.13.

Assuming that x_i and y_i are the numbers of the attributes of the objects under comparison, which can be mapped onto each other because they belong to the same semantically homogeneous category i and that $x_i \geq y_i$, the worst case time complexity of the *SelectOptimalIsomorphism* algorithm is $O_t(\sum_i \frac{x_i!}{(x_i-y_i)!})$.

For each semantically homogeneous category of attributes i , the number of all the possible isomorphisms between the x_i attributes of object x and the y_i attributes of object y equals to $\frac{x_i!}{(x_i-y_i)!}$. This is because these isomorphisms can be generated by first creating all the $\frac{x_i!}{y_i!(x_i-y_i)!}$ possible combinations of the x_i and the y_i attributes of the objects, and then taking all the $y_i!$ possible permutations of each of these combinations.

Furthermore, since the categories i are pairwise disjoint (see the criterion of the semantic homogeneity in chapter 3), the total computational cost of generating a single optimal isomorphism, equals the sum of the costs of generating the partial optimal isomorphisms for each of the categories i (i.e. $O_t(\sum_i \frac{x_i!}{(x_i-y_i)!})$).

An upper bound for this complexity is given by the following theorem.

Theorem 5.1: *The worst case time complexity of the SelectOptimalIsomorphism algorithm considering two objects x and y with n_x and n_y attributes, respectively, is bounded by:*

$$O_t(n!), \quad n = \max\{n_x, n_y\}$$

if $n_x, n_y > 2$ and for all the semantically homogeneous categories i $x_i \geq y_i$ where x_i, y_i are the numbers of the attributes of objects x, y belonging to category i .

PROOF: It is sufficient to prove the following two inequalities:

1. for each i it holds $\frac{x_i!}{(x_i-y_i)!} \leq x_i!$ if $x_i, y_i \in Z^+$, and $x_i \geq y_i$ (I); and,

$$2. \sum_{i=1}^k x_i! \leq (\sum_{i=1}^k x_i)! \quad (\text{II})$$

since according to them, we have that,

$$\sum_i \frac{x_i!}{(x_i - y_i)!} \leq \sum_i x_i! \leq (\sum_i x_i)! = n_x!$$

• Inequality (I) is proven as follows:

For any fixed x_i , the quantity $\frac{x_i!}{(x_i - y_i)!}$ becomes maximum when $(x_i - y_i)!$ becomes minimum.

However, since the factorial function $f(x) = x!$ is a monotonically increasing function over the set Z^+ , the quantity $(x_i - y_i)!$ becomes minimum when $(x_i - y_i) = 0$. Furthermore, by assumption $x_i \geq y_i$. Thus, $(x_i - y_i) = 0$ when $x_i = y_i$.

$$\text{Then, } \frac{x_i!}{(x_i - y_i)!} = \frac{x_i!}{0!} = x_i!$$

$$\text{Therefore, } \frac{x_i!}{(x_i - y_i)!} \leq x_i!$$

• Inequality (II) is proven by induction as follows:

i) for $k = 1$:

$$\sum_{i=1}^1 x_i! = x_1! = (\sum_{i=1}^1 x_i)!$$

ii) for $k = 2$:

Since $n_x > 2$ we have that $x_1 + x_2 > 2$ and thus it is impossible that $x_1 = x_2 = 0$. Let us assume, without loss of generality that $x_2 > 0$. Then, we have:

$$\sum_{i=1}^2 x_i! \leq (\sum_{i=1}^2 x_i)! \iff x_1! + x_2! \leq (x_1 + x_2)! \iff 1 + \frac{x_2!}{x_1!} \leq \frac{(x_1 + x_2)!}{x_1!} \quad (\text{III})$$

If $x_2 = x_1$, (III) becomes

$$2 \leq \frac{(2x_2)!}{x_2!} \iff 2 \leq \frac{(2x_2)(2x_2-1) \cdots (x_2)!}{x_2!} \iff 2 \leq (2x_2)(2x_2-1) \cdots (x_2 + 1)$$

which is true since $x_2 \geq 1$.

If $x_2 > x_1$, (III) becomes

$$1 + \frac{(x_2)(x_2-1) \cdots (x_1+1)(x_1)!}{x_1!} \leq \frac{(x_2+x_1)(x_2+x_1-1) \cdots (x_1+1)(x_1)!}{x_1!} \iff$$

$$1 + (x_2)(x_2-1) \cdots (x_1+1) \leq (x_2+x_1)(x_2+x_1-1) \cdots (x_1+1) \iff$$

$$2 \leq (x_2+x_1) \cdots (x_2+1)$$

which is true since $x_1+x_2 \geq 2$.

iii) Assume that (III) holds for $k = m$, that is:

$$\sum_{i=1}^m x_i! = (\sum_{i=1}^m x_i)!$$

Then for $k = m + 1$ we have:

$$\sum_{i=1}^{m+1} x_i! = x_1! + x_2! + \cdots + x_m! + x_{m+1}! = (x_1! + x_2! + \cdots + x_m!) + x_{m+1}!$$

However by our assumption about the validity of (II) for $k=m$

$$x_1! + x_2! + \cdots + x_m! + x_{m+1}! \leq (x_1 + x_2 + \cdots + x_m)! + x_{m+1}!$$

and the proven validity of (II) for $k=2$, we have

$$(x_1 + x_2 + \cdots + x_m)! + x_{m+1}! \leq (x_1 + x_2 + \cdots + x_m + x_{m+1})! = (\sum_{i=1}^{m+1} x_i)!$$

Therefore, (II) is also true for $k=m+1$.

□

The worst case space complexity of this algorithm is $O_s(\sum_i \frac{x_i!}{(x_i-y_i)!})$. This is because the

algorithm proceeds incrementally by generating all the $\sum_i \frac{x_i!}{(x_i-y_i)!}$ possible isomorphisms

in order to select the optimal one and for each of them it is necessary to keep the mappings between the attributes. Like the time complexity, the space complexity of the algo-

gorithm is also bounded by $O_s(n!)$, where $n = \max\left\{\sum_i x_i, \sum_i y_i\right\}$.

5.2.10 The AttributionDistance Algorithm

The algorithm computing the attribution distance between objects proceeds through three basic stages.

First, it retrieves all the attributes of the objects(i.e. both their direct and inherited attributes) and checks which of them could be mapped onto each other, according to the criterion of semantic homogeneity. At the same stage, it also computes the distances between all the pairs of the semantically homogeneous attributes(i.e the invocation of the *PossibleMappings* algorithm).

Then, it incrementally generates all the possible isomorphisms between the semantically homogeneous attributes and selects the isomorphism with the minimum distance(i.e the invocation of the *SelectOptimalIsomorphism* algorithm).

Finally, it adds the distance of the selected optimal isomorphism with the partial distances due to the semantically heterogeneous attributes of the involved objects(i.e. attributes of an object, which are mapped onto no attribute of the other). These distances equal the squares of the saliences of the semantically heterogeneous attributes.

The algorithm is specified in figure 5.14.

The worst case time complexity of this algorithm is

$$O_i(\sum_i x_i \sum_i y_i K_{map} + \sum_i x_i K_{sal} + \sum_i x_i y_i K_D + \sum_i \frac{x_i!}{(x_i - y_i)!} + \sum_i y_i K_{sal})$$

The factor $\sum_i x_i \sum_i y_i K_{map} + \sum_i x_i K_{sal} + \sum_i x_i y_i K_D$ is the cost of invoking the *PossibleMappings* algorithm(see section 5.2.8). The factor $\sum_i \frac{x_i!}{(x_i - y_i)!}$ is the cost of the invocation of the *SelectOptimalIsomorphism* algorithm(see section 5.2.9). Finally, the factor $\sum_i y_i K_{sal}$ refers to the cost of estimating the salience of the attributes of #s2 which were not semantically homogeneous to any of the attributes of #s1 and thus, their salience cannot be derived from the salience of any of those attributes(see section 3.3.4 in chapter 3). K_{map} , K_{sal} and K_D have been defined in section 5.2.8.

Subject to the boundaries of these three costs, discussed in sections 5.2.8 and 5.2.9, the complexity of the *AttributionDistance* algorithm is bounded by

ALGORITHM: AttributionDistance($\beta_4, V_s, \#s1, \#s2, d, I$)

INPUT: β_4 /* a real number greater than 0 used for the homographic transformation of absolute attribution distances */

$\#s1, \#s2$ /* the unique identifiers of the objects under comparison */

V_s /* a set used to store identifiers of objects currently being under comparison, so as to preclude the comparison of attributes having such objects as values (i.e. cyclic attributes), see definition 3.34 in chapter 3 */

OUTPUT: d /* the attribution distance d_4 of the objects identified by $\#s1$ and $\#s2$ */

I /* the optimal isomorphism between the attributes of the objects identified by $\#s1$ and $\#s2$; it is a list of triplets ($\#i, \#j, \text{Dij}$) denoting maps between $\#i$ and $\#j$ and their pairwise distance Dij */

STEPS:

A1 = get_all_attributes($\#s1$);

A2 = get_all_attributes($\#s2$);

PossibleMappings($V_s, A1, A2, \text{PM}$);

/* PM is a list storing the the salience of the attributes of $\#s1$, their possible maps of onto the attributes of $\#s2$ and the distances of each mapping pair, structured as as illustrated in figure 5.11 */

SelectOptimalIsomorphism(PM, I, d);

for each $\#a$ in A1 and ($\text{not}(\#a \in I)$) **do**

$d = d + \text{sal}(\#a)^2$;

endfor

for each $\#a$ in A2 and ($\text{not}(\#a \in I)$) **do**

get_all_classes($\#a, C$); max_sal = 0;

for each $\#i$ in C **do**

GetOriginalClass($\#i, \#o$); EvaluateSalience($\#o, \text{sal}$);

if sal > max_sal **then** max_sal = sal **endif**

endfor

$d = d + \text{sal}^2$;

endfor

$d = \beta_4 d / (\beta_4 d + 1)$;

Figure 5.14: AttributionDistance Algorithm

$$O_t(N_1 N_2 (K_{map} + K_D) + (N_1 + N_2) K_{sal} + N!)$$

where

- $N_1 = \sum_i x_i$ (i.e. the number of attributes of $\#s1$);
- $N_2 = \sum_i y_i$ (i.e. the number of attributes of $\#s2$); and,
- $N = \max\{N_1, N_2\}$.

The worst case space complexity of the *AttributionDistance* algorithm is determined by the worst case complexity of the *SelectOptimalIsomorphism* and thus it equals to

$O_s(\sum_i \frac{x_i!}{(x_i - y_i)!})$. This complexity is bounded according to theorem 5.1.

5.2.11 The ObjectDistance Algorithm

The *ObjectDistance* algorithm computes the overall distance between two objects and the optimal isomorphism between their attributes.

The algorithm first checks, whether the objects under comparison are identical or not. If they are identical, it simply reports that their distance is 0. Otherwise, it checks whether they are both individuals or attributes of the same instantiation level. If they are not, it reports that similarity analysis is invalid by the principle of *ontological uniformity*(see chapters 2 and 3). Similarity between objects which satisfy this principle is analyzed.

This analysis includes the computation of their classification and attribution distances, as well as the computation of their generalization distance, if the involved objects are both classes(for token objects the generalization distance is not defined).

The algorithm is specified in figure 5.15.

Worst Case Overall Time Complexity. The *ObjectDistance* algorithm proceeds recursively along the attribution of objects, while computing their attribution distance because the distances between the object/values of their attributes are necessary in computing the distances between their attributes themselves. Thus, its overall time complexity must be expressed through a recursive equation. According to the analysis about the time complexities of its component algorithms, the overall time complexity of the *ObjectDistance* algorithm, at each level of recursion x is given by the following formula:

$$T_x = K_{in} + K_{isa} + \left(\sum_i x_i + \sum_i y_i\right)K_{sal} + \left(\sum_i x_i\right)\left(\sum_i y_i\right)K_{map} + \left(\sum_i x_i y_i + 1\right)T_{x-1} + \sum_i \frac{x_i!}{(x_i - y_i)!} \quad (5.1)$$

where

- K_{in} is the cost of estimating the classification distance between the involved objects(see the time complexity of the *ClassificationDistance* algorithm);
- K_{isa} is the cost of estimating the generalization distance between the involved objects(see the time complexity of the *GeneralizationDistance* algorithm);
- K_{map} is the cost of testing the semantic homogeneity of a pair of attributes (see the analysis of the time complexity of the *PossibleMappings* algorithm); and,

- K_{sal} is the cost of estimating the salience of an attribute (see the time complexity of the *EvaluateSalience* algorithm).

Formula 5.1 accumulates for each recursion step x the costs of estimating the classification(i.e. the factor K_{in}), generalization(i.e. the factor K_{isa}) and attribution distances(i.e. the factor $(\sum_i x_i \sum_i y_i)K_{map} + (\sum_i x_i + \sum_i y_i)K_{sal} + (\sum_i x_i y_i + 1)T_{x-1} + \sum_i \frac{x_i!}{(x_i - y_i)!}$).

This formula can be rewritten as:

$$T_x = AT_{x-1} + B \quad (5.2)$$

where

$$A = \sum_i x_i y_i + 1 \quad (5.3)$$

and

$$B = K_{in} + K_{isa} + \sum_i x_i \sum_i y_i K_{map} + (\sum_i x_i + \sum_i y_i)K_{sal} + \sum_i \frac{x_i!}{(x_i - y_i)!} \quad (5.4)$$

Assuming that, factors A and B do not have different values at the different levels of recursion x , formula 5.2 is a *linear first order difference equation*, whose general solution is given by the expression(see [YK80]):

$$T_x = B \frac{A^x - 1}{A - 1} + CA^x \quad (5.5)$$

where

$$C = T_0$$

T_0 is the cost of similarity analysis between objects having no attributes. We define this cost to be equal to $K_{in} + K_{isa}$. This implies that the analysis of similarity between objects having no attributes depends solely on the estimation of their classification and generalization distances. We can also control the maximum depth of the recursion in analysis, in a similar manner. By setting in advance a maximum allowed level of recursion x_m , the *ObjectDistance* algorithm will proceed along the attribution graphs of objects up to a depth x_m and then, it will estimate the overall distance of the objects at that level, solely from their classification and generalization distances. In both these cases, the factor C in formula 5.5 will be equal to $K_{in} + K_{isa}$.

ALGORITHM: ObjectDistance($V_s, \#s1, \#s2, I, D$)

INPUT: V_s /* a set used to store identifiers of objects currently being under comparison, so as to preclude the comparison of attributes having such objects as values (i.e. cyclic attributes); when the algorithm is initially called V_s is empty */

$\#s1, \#s2$ /* two identifiers of objects to be compared */

OUTPUT: I /* the optimal isomorphism between the attributes of the input objects */

D /* the overall distance of the input objects */

STEPS:

set parameters $\beta_2, \beta_3, \beta_4$; /* parameters for homographic transformations of absolute distances */

$I =$ empty list; /* I is a list of pairs $(\#i, \#j, dij)$ where $\#i, \#j$ are attributes of $\#s1$ and $\#s2$, respectively and dij their distance */

if $\#i = \#j$ **then**

$D = 0$;

else

if $\#i$ and $\#j$ have the same instantiation level **then**

if $\#i$ and $\#j$ are both individuals **then**

$V_s = V_s \cup \{\#i, \#j\}$; $d_1 = 1$;

$ClassificationDistance(\beta_2, \#i, \#j, d_2)$; $AttributionDistance(V_s, \beta_4, \#i, \#j, d_4, I)$;

if $\#i$ and $\#j$ are classes **then**

$GeneralizationDistance(\beta_3, \#i, \#j, d_3)$;

$D = (d_1^2 + d_2^2 + d_3^2 + d_4^2 + 2d_2d_3 + 2d_2d_4 + 2d_3d_4)^{1/2}$;

else

$D = (d_1^2 + d_2^2 + d_4^2 + 2d_2d_4)^{1/2}$;

endif

else

if $\#i$ and $\#j$ are both attributes **then**

$d_1 = 1$;

$ClassificationDistance(\beta_2, \#i, \#j, d_2)$; $AttributionDistance(V_s, \beta_4, \#i, \#j, d_4, I)$;

$x_1 = o(\#i).TO$; $x_2 = o(\#j).TO$; /* x_1, x_2 are the object/values of $\#i, \#j$ */

if $x_1 \in V_s$ or $x_2 \in V_s$ **then** /* estimation of distance between cyclic attributes */

if $x_1 = x_2$ **then** $d'_1 = 0$; **else** $d'_1 = 1$; **endif**

$ClassificationDistance(\beta_2, x_1, x_2, d'_2)$; $GeneralizationDistance(\beta_3, x_1, x_2, d'_3)$;

$D' = (d'_1{}^2 + d'_2{}^2 + 36d'_3{}^2 + 12d'_2d'_3)^{1/2}$ /* D' is the distance of the values of the attributes */

else

$ObjectDistance(V_s, x_1, x_2, I, D')$; /* D' is the distance of the values of the attributes */

endif

if $\#i$ and $\#j$ are classes **then**

$GeneralizationDistance(\beta_3, \#i, \#j, d_3)$;

$D = (d_1^2 + d_2^2 + 36d_3^2 + d_4^2 + D'^2 + 2d_2d_4 + 12d_2d_3d_4)^{1/2}$;

else

$D = (d_1^2 + d_2^2 + d_4^2 + D'^2 + 2d_2d_4)^{1/2}$;

endif

else report invalid analysis request;

endif

endif

else report invalid analysis request;

endif

del($V_s, \#i$); del($V_s, \#j$);

endif

Figure 5.15: The ObjectDistance Algorithm

Notice that our assumption about A,B is valid if we replace them with their maximum possible values regarding all the object-pairs involved in the estimation of the overall distance of an initial pair of objects. This is because factors A,B depend on various parameters of the particular objects being compared at each step of analysis, as shown in the preceding sections. Thus, formula 5.5 gives a conservative boundary of the worst case time complexity of either unrestricted similarity analysis for objects with attribution graphs having a maximum depth equal to x or analysis restricted to a maximum depth of x.

Overall Worst Space Complexity. The space complexity of the *ObjectDistance* algorithm is $O_s(\max\left\{(n_1 + n_2), (n_3 + n_4), \sum_i \frac{x_i!}{(x_i - y_i)!}\right\})$

where

- n_1 and n_2 are the numbers of the classes of the input objects ;
- n_3 and n_4 are the numbers of the superclasses of the input objects ; and,
- x_i and y_i are the numbers of the attributes of the input objects, which belong to the semantically homogeneous category i.

The factors $(n_1 + n_2)$, $(n_3 + n_4)$ and $\sum_i \frac{x_i!}{(x_i - y_i)!}$ are the worst case space complexities of the computations of the classification, generalization and attribution distances, respectively. These complexities are discussed in the sections specifying the relevant algorithms. Due to the bounding of the factor $\sum_i \frac{x_i!}{(x_i - y_i)!}$ by theorem 5.1, the worst case space complexity is also bounded by $O_s(\max\left\{(n_1 + n_2), (n_3 + n_4), N!\right\})$ where N is the maximum number of attributes of the involved objects (i.e. $N = \max\left\{N_1, N_2\right\}$).

5.3 Further Complexity Issues

5.3.1 The Attribution Distance as an Assignment Problem

The estimation of the distance over the attribution of objects can be expressed as a weighted bipartite graph matching problem (also referred to as the *assignment problem*[PS82]).

Since semantic homogeneity is an equivalence relation(see theorem 3.6 in chapter 3), it decomposes each of the sets of the attributes of two objects $\#i$ and $\#j$, $A_{\#i}$ and $A_{\#j}$, into the following two partitions(see cases (a) and (b) in figure 5.16):

$$A_{\#i}^{(1)}, \dots, A_{\#i}^{(n)} \text{ where } A_{\#i}^{(p)} \cap A_{\#i}^{(q)} = \emptyset \quad \forall \quad p, q \quad 1 \leq p, q \leq n$$

$$A_{\#j}^{(1)}, \dots, A_{\#j}^{(m)} \text{ where } A_{\#j}^{(p)} \cap A_{\#j}^{(q)} = \emptyset \quad \forall \quad p, q \quad 1 \leq p, q \leq m$$

Let $E_{\#i}$ and $E_{\#j}$ be two sets with elements all the sets $A_{\#i}^{(p)}$ and all the sets $A_{\#j}^{(q)}$, respectively.

Then, we can define an isomorphism R between $E_{\#i}$ and $E_{\#j}$ as:

$$R = \left\{ (X, Y) \mid (X \in E_{\#i}) \text{ and } (Y \in E_{\#j}) \text{ and } \exists x, y : (x \in X) \text{ and } (y \in Y) \text{ and } sh(x, y) \right\}$$

R is an isomorphism since for any two pairs of elements (X, Y) , (X, Z) and (W, Z) , (V, Z) belonging to it, $Y=Z$ and $W=Z$ (see case (b) in figure 5.16).

In fact, suppose that $Y \neq Z$ and in particular that there exists an element y of Y that does not also belong to Z . Notice that y is semantically homogeneous to every element x of X (i.e. $sh(y, x) \forall x \in X$). However, since (X, Z) also belongs to R, each element x of X is semantically homogeneous to all the elements of Z (i.e. $sh(x, z) \forall z \in Z$). Then, y is semantically homogeneous to z and thus it must be member of Z . This contradicts our initial assumption that y does not belong to Z and therefore, it must be $Y \subseteq Z$. Similarly it can be proven that $Z \subseteq Y$, thus $Y=Z$. The proof that $W=V$ is identical.

For each element $e=(A_{\#i}^{(e)}, A_{\#j}^{(e)})$ of R, we can define a complete bipartite graph $G^{(e)}$, with even number of nodes, as:

$$G^{(e)} = (A_{\#i}^{(e)}, A_{\#j}^{(e)}, E^{(e)})$$

where

$$A_{\#i}^{(e)} = \begin{cases} A_{\#i}^{(e)} & \text{if } |A_{\#i}^{(e)}| \geq |A_{\#j}^{(e)}| \\ A_{\#i}^{(e)} \cup D_{\#i}^{(e)}, D_{\#i}^{(e)} = \{\#d_{i1}^{(e)}, \dots, \#d_{in}^{(e)}\} & n = |A_{\#j}^{(e)}| - |A_{\#i}^{(e)}| \text{ otherwise} \end{cases}$$

$$A_{\#j}^{(e)} = \begin{cases} A_{\#j}^{(e)} & \text{if } |A_{\#j}^{(e)}| \geq |A_{\#i}^{(e)}| \\ A_{\#j}^{(e)} \cup D_{\#j}^{(e)}, D_{\#j}^{(e)} = \{\#d_{j1}^{(e)}, \dots, \#d_{jm}^{(e)}\} & m = |A_{\#i}^{(e)}| - |A_{\#j}^{(e)}| \text{ otherwise} \end{cases}$$

and

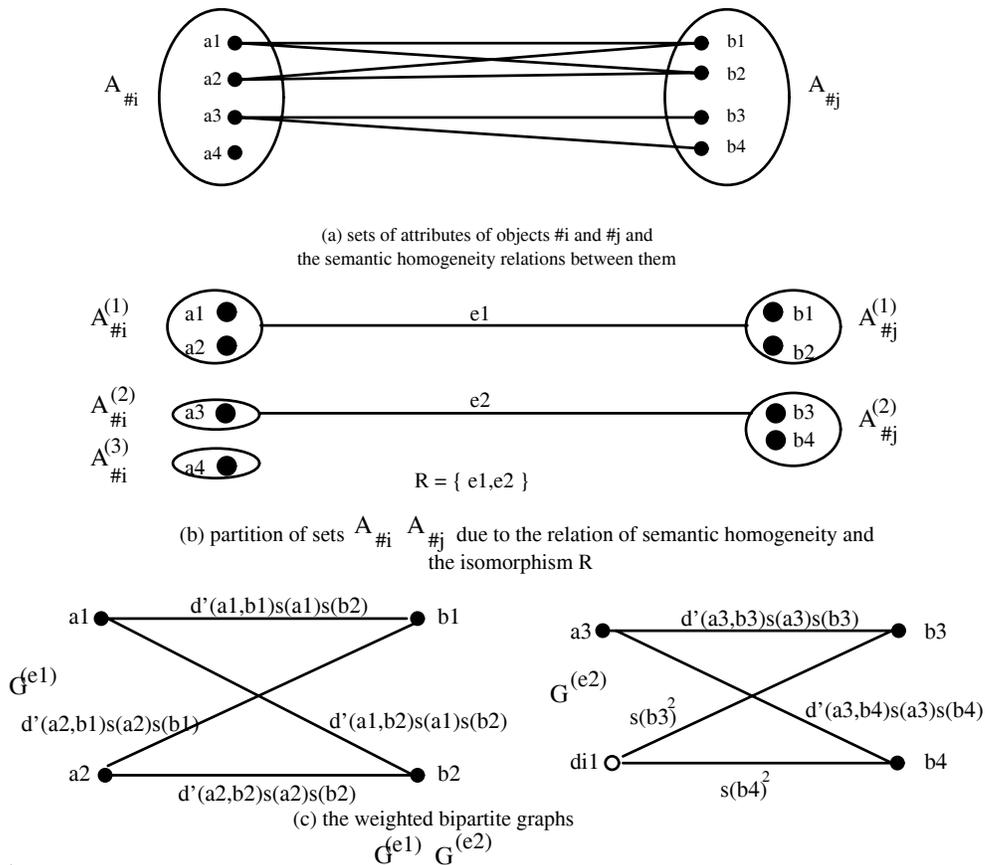


Figure 5.16: Bipartite Graphs Mapping Semantically Homogeneous Attributes

$$E^{(e)} = A'_{\#i}^{(e)} \times A'_{\#j}^{(e)}$$

In essence, these definitions augment the set of the pair $(A_{\#i}^{(e)}, A_{\#j}^{(e)})$ having the fewer attributes with special identifiers referring to dummy attributes, so as both $A'_{\#i}^{(e)}$ and $A'_{\#j}^{(e)}$ will have equal numbers of elements. Also $A'_{\#i}^{(e)} \cap A'_{\#j}^{(e)} = \emptyset$ since the sets $A_{\#i}^{(e)}$ and $A_{\#j}^{(e)}$ include attributes from different objects (see case (c) in figure 5.16).

The set of edges $E^{(e)}$ is partitioned into the sets

$$E_{ho}^{(e)} = A_{\#i}^{(e)} \times A_{\#j}^{(e)}, \quad E_{di}^{(e)} = A_{\#i}^{(e)} \times D_{\#j}^{(e)} \quad \text{and} \quad E_{dj}^{(e)} = D_{\#i}^{(e)} \times A_{\#j}^{(e)}$$

In words,

- $E_{ho}^{(e)}$ includes the edges expressing all the possible mappings between the semantically homogeneous attributes in $A_{\#i}^{(e)}$ and $A_{\#j}^{(e)}$;

- E_{di} includes the edges connecting the attributes of $\#i$ with the dummy attributes of $\#j$, which in fact express mappings of the former attributes to *null* attributes ; and,
- E_{dj} includes the edges connecting the attributes of $\#j$ with the dummy attributes of $\#i$, which in fact express mappings to the former attributes to *null* attributes.

Furthermore, each edge $(\#v, \#u)$ in $E^{(e)}$ is assigned a weight $c_{\#v\#u}$ defined as

$$c_{\#v\#u} = \begin{cases} d'(\#v, \#u) s(\#v) s(\#u) & \text{if } (\#v, \#u) \in E_{ho}^{(e)} \\ s(\#v)^2 & \text{if } (\#v, \#u) \in E_{di} \\ s(\#u)^2 & \text{if } (\#v, \#u) \in E_{dj} \end{cases}$$

The factors $d'(\#v, \#u)$, $s(\#v)$, $s(\#u)$ are defined by definition 3.34 in chapter 3.

Then, the weighted bipartite graph matching problem is to find a matching $M^{(e)}$ (i.e. a subset of $E^{(e)}$) with the minimum possible sum of weights.

In an algebraic formulation, let the variables $x_{\#v\#u}$ ($\#v \in A'_{\#i}^{(e)}$ and $\#u \in A'_{\#j}^{(e)}$) denote whether or not an edge belongs to a matching of $G^{(e)}$ by taking the values 1 and 0, respectively. The weighted bipartite graph matching problem is to find an assignment to the variables $x_{\#v\#u}$, minimizing the objective function

$$\sum_{\#v \in A'_{\#i}^{(e)}, \#u \in A'_{\#j}^{(e)}} x_{\#v\#u} c_{\#v\#u} \quad (\text{F1})$$

while satisfying the constraints

$$\sum_{\#u \in A'_{\#j}^{(e)}} x_{\#v\#u} = 1 \quad \forall \#v \in A'_{\#i}^{(e)} \quad (\text{c1})$$

$$\sum_{\#v \in A'_{\#i}^{(e)}} x_{\#v\#u} = 1 \quad \forall \#u \in A'_{\#j}^{(e)} \quad (\text{c2})$$

and

$$x_{\#v\#u} = 0 \text{ or } 1 \quad \forall \#v, \#u : \#v \in A'_{\#i}^{(e)} \text{ and } \#u \in A'_{\#j}^{(e)} \quad (\text{c3})$$

The constraint c1(c2) together with c3 guarantee that elements of $A'_{\#i}^{(e)}$ ($A'_{\#j}^{(e)}$) can be mapped onto only one element of $A'_{\#j}^{(e)}$ ($A'_{\#i}^{(e)}$). Also the minimization of function F1 guarantees that the isomorphism between the elements of $A'_{\#i}^{(e)}$ and $A'_{\#j}^{(e)}$ will have the lowest possible overall attribute distance, due to the particular assignment of factors $c_{\#v\#u}$. It is known that even if (c3) is relaxed to $x_{\#v\#u} \geq 0$, the resulting linear programming problem is of a special form that guarantees the existence of integer (therefore, 0 or 1) optimal

solutions.

Notice that any attribute in $A_{\#i}^{(e)}$ can only be mapped onto one attribute of $A_{\#j}^{(e)}$ (and vice versa) or to the null attribute. Also the weights of the edges in $E^{(e)}$ are exactly the distance factors between semantically homogeneous attributes (for edges in $E_{ho}^{(e)}$) or unmapped attributes (for edges in $E_{di}^{(e)} \cup E_{dj}^{(e)}$) defined by definition 3.34 in chapter 3. Thus, the optimal matching of the bipartite graph $G^{(e)}$ will also be the optimal isomorphism between the attributes $A_{\#i}^{(e)}$ and $A_{\#j}^{(e)}$.

Therefore, solving the bipartite weighted graph matching problem for all the graphs $G^{(e)}$ defined by R and taking the union of the obtained optimal matchings, we obtain the optimal isomorphism between the attributes of the objects #i and #j. Attributes belonging to those sets of $E_{\#i}$ and $E_{\#j}$ which are not included in R cannot be mapped to any attribute of the other object and their contribution to the attribution distance is fixed (i.e. equal to the square of their saliences). These factors must be added to the sum of the minimal distances obtained by solving the bipartite weighted graph matching problems for the graphs $G^{(e)}$.

The expression of the estimation of the attribution distance as weighted bipartite graph matching problem implies that it can be computed by algorithms more efficient than the branch-and-bound algorithm of our current implementation in the worst case. A traditional such algorithm is the *Hungarian Method*, which has a worst case time complexity equal to $O_i(n^3)$, for graphs with $2n$ nodes [PS82].

Using this algorithm, the attribution distance problem can be computed in at most $N_{\#i}^3 + N_{\#j}^3$ steps, where $N_{\#i}$ and $N_{\#j}$ are the total numbers of attributes of objects #i and #j (i.e. $N_{\#i} = \sum_e |A_{\#i}^{(e)}|$ and $N_{\#j} = \sum_e |A_{\#j}^{(e)}|$). This is because, the total cost of solving the assignment problems for all the graphs $G^{(e)}$ is bounded as follows:

$$\sum_e \max(x_e^3, y_e^3) \leq \sum_e x_e^3 + \sum_e y_e^3$$

where $x_e = |A_{\#i}^{(e)}|$ and $y_e = |A_{\#j}^{(e)}|$.

Hence, the factor B in formulas 5.2 and 5.5 becomes

$$B = K_{in} + K_{isa} + \sum_i x_i \sum_i y_i K_{map} + (\sum_i x_i + \sum_i y_i) K_{sal} + \sum_i x_i^3 + \sum_i y_i^3 \quad (5.6)$$

Therefore, the worst case time complexity of the overall distance between two objects,

expressed by formula 5.5, becomes exponential only to the depth x of recursively estimating distances between objects in the transitive closures of their attribution graphs.

5.3.2 Average Time Complexity

This section analyzes the average case time complexity of the *ObjectDistance* algorithm, when the attribution distance between objects is estimated by the branch-and-bound algorithm of our current implementation. Our analysis concentrates on this distance since evidently by formula 5.1, the only combinatorial factor in the overall complexity (i.e. the factor $\sum_i \frac{x_i!}{(x_i - y_i)!}$) depends on the numbers of attributes of the involved objects in each of the semantically homogeneous categories i . These numbers affect the base of the exponential factor A^x (see formula 5.3). The other factors of the overall computational cost (i.e. the factors K_{in} , K_{isa} , K_{map} and K_{sal}) introduce only polynomial complexities (see the analysis of the relevant algorithms) and thus their effect on the overall cost is less significant.

A conservative approximation to the average overall time complexity of the algorithm can be obtained by considering the factors A, B and C in the formula 5.5 as random variables and consequently substituting them by their expected values $E(A)$, $E(B)$ and $E(C)$. Then the average complexity for objects with attribution graphs, having a maximum depth of x , will be given by the formula:

$$T_x = E(B) \frac{E(A)^x - 1}{E(A) - 1} + E(C) E(A)^x \quad (5.7)$$

where

$$E(C) = E(T_0) = E(K_{in} + K_{isa}) = E(K_{in}) + E(K_{isa})$$

The polynomial upper bounds of factors K_{in} and K_{isa} do not make necessary the further elaboration of their expected values.

The expected values $E(A)$ and $E(B)$ can be estimated by considering the numbers of the attributes of two objects x and y , in any of the semantically homogeneous categories i (i.e. x_i and y_i) as random variables and hypothesizing their distributions.

In the subsequent analysis, x_i and y_i are assumed to follow the Poisson distribution with the same average value L . Thus, the probability of having z attributes of an arbitrary object, belonging to some semantically homogeneous category is given by the formula:

$$f(z) = \frac{e^{-L} L^z}{z!}, z=0,1,2,\dots$$

Certain properties of the Poisson distribution indicate that it is not unreasonable to hypothesize it as the distribution of the number of attributes in a semantically homogeneous category. First, it is a discrete distribution, which may theoretically take any value in the set of the natural numbers. However, the probabilities of values, which are greater than four times its average value are very close to 0. Thus, very large values are not probable.

When the average value L is small (i.e. less than 3) the distribution is left skew and tends to become symmetric for larger average values. Thus, it can approximate a wide range of unimodal distributions of random discrete variables, except for right skew ones. Considering the empirical distribution of the number of the attributes in semantically homogeneous categories, the only possible case of it being right skew is when the attributes of objects are all classified only under the general class of attributes *Attribute* (see chapter 3) and, furthermore, the number of the attributes of an object has a right skew distribution. Such cases are not expected when descriptions of objects are devised according to specific meta models abstracting different types of relations in various domains.

Furthermore, x_i and y_i can be assumed to be independent variables. Indeed, the classification of the attributes of an object under certain classes of attributes, which consequently determines their category of semantic homogeneity, does not depend on the classification of the attributes of any other object, in general.

It is known by probability theory [Drake67], that if a set of values $x_i, i = 1, 2, \dots, n$ of a random variable X (where n is a random variable itself with expected value $E(n)$) are independent and distributed according to some distribution with average value $E(X)$, their sum has an expected value, given by the expression:

$$E\left(\sum_i^n x_i\right) = E(n)E(X)$$

Consequently, if d is the expected number of the distinct semantically homogeneous categories i and L is the expected number of the attributes of an object in some category i , the expected value of the number of attributes of an object, L_o , is given by the expression:

$$L_o = dL$$

This relation is important for estimating the average time complexity of the similarity analysis model, since estimates about the expected values L and L_o can be obtained unlike estimates of the expected number d of distinct categories of semantic homogeneity, which depend on the particular pairs of objects, whose similarity could analyzed.

Estimation of E(A). The expected value $E(A)$ has the following upper bound:

$$E(A) = L_o L + 1 \quad (5.8)$$

where L_o is the average number of attributes of an object and L is the average number of attributes of an object, in an arbitrary semantically homogeneous category.

Formula 5.8 is derived as it follows:

$$E(A) = E\left(\sum_{i=1}^d x_i y_i + 1\right) = \sum_{i=1}^d E(x_i y_i) + 1 = \sum_{i=1}^d E(x_i) E(y_i) + 1 = dL^2 + 1 = L_o L + 1$$

Estimation of E(B). The expected value $E(B)$, where B is given by formula 5.4 is:

$$E(B) = E_{in} + E_{isa} + 2L_o E_{sal} + L_o^2 E_{map} + L_o L e^{L^2-L} \quad (5.9)$$

where

- E_{map} is the expected value of testing the semantic homogeneity between two attributes ;
 - E_{in} is the expected value of estimating the classification distance between two objects ;
 - E_{isa} is the expected value of estimating the generalization distance between two objects ;
- and,
- E_{sal} is the expected value of estimating the salience of an attribute.

As discussed in the previous sections, all these costs are polynomial in the worst case. Hence, we will not elaborate further in their derivation. Instead, we will concentrate on the factors of formula 5.4, which introduce an exponential complexity.

According to that formula, we have:

$$E(B) = E(K_{in}) + E(K_{isa}) + E\left(\sum_i x_i \sum_i y_i K_{map}\right) + E\left(\left(\sum_i x_i + \sum_i y_i\right) K_{sal}\right) + E\left(\sum_i \frac{x_i!}{(x_i - y_i)!}\right) \quad (5.10)$$

Certain additional independency assumptions allow the estimation of $E(B)$.

First, the variables X (where $X = \sum_i x_i$) and Y (where $Y = \sum_i y_i$) can be assumed independent, since the attribution of an object does not depend on the attribution of another.

Second, the test of the semantic homogeneity of two attributes depends on their classification. Classifications of specific attributes are not related to the total number of attributes of their owning objects. Thus, the cost of checking the semantic homogeneity of two attributes(i.e. K_{map}) does not depend on the variables X and Y .

Third, the estimation of salience depends on the classes of the involved attributes, which are also not related to X and Y . Therefore, the cost of estimating the salience of attributes (i.e. K_{sal}) does not depend on the variables X and Y .

These independence assumptions imply that 5.10 is equal to:

$$E(K_{in}) + E(K_{isa}) + E(\sum_i x_i)E(\sum_i y_i)E(K_{map}) + (E(\sum_i x_i) + E(\sum_i y_i))E(K_{sal}) + E(\sum_i \frac{x_i!}{(x_i-y_i)!})$$

The expected values $E(\sum_i x_i)$ and $E(\sum_i y_i)$ are both equal to L_o or, equivalently, to dL .

Also, the expected value $E(\sum_i \frac{x_i!}{(x_i-y_i)!})$ can be derived as it follows:

$$E(\sum_i \frac{x_i!}{(x_i-y_i)!}) = \sum_i E(\frac{x_i!}{(x_i-y_i)!}) \quad (5.11)$$

However,

$$E(\frac{x_i!}{(x_i-y_i)!}) = \sum_{x_i=0}^{\infty} \sum_{y_i=0}^{x_i} \frac{x_i!}{(x_i-y_i)!} f(x_i, y_i)$$

Thus, due to the assumed independence of x_i and y_i , we have:

$$\begin{aligned} E(\frac{x_i!}{(x_i-y_i)!}) &= \sum_{x_i=0}^{\infty} \sum_{y_i=0}^{x_i} \frac{x_i!}{(x_i-y_i)!} f(x_i)f(y_i) = \sum_{x_i=0}^{\infty} \sum_{y_i=0}^{x_i} \frac{x_i!}{(x_i-y_i)!} \frac{e^{-L} L^{x_i}}{x_i!} \frac{e^{-L} L^{y_i}}{y_i!} = \\ &= \sum_{x_i=0}^{\infty} \sum_{y_i=0}^{x_i} \frac{x_i!}{y_i!(x_i-y_i)!} \frac{e^{-2L} L^{x_i} L^{y_i}}{x_i!} = \sum_{x_i=0}^{\infty} \sum_{y_i=0}^{x_i} \binom{x_i}{y_i} \frac{e^{-2L} L^{x_i} L^{y_i}}{x_i!} \end{aligned}$$

However, by the *polynomial theorem*(see [YK80]):

$$\sum_{y_i=0}^{x_i} \binom{x_i}{y_i} \frac{e^{-2L} L^{x_i} L^{y_i}}{x_i!} = e^{-2L} \frac{L^{x_i}}{x_i!} \sum_{y_i=0}^{x_i} \binom{x_i}{y_i} L^{y_i} = e^{-2L} \frac{L^{x_i}}{x_i!} (1+L)^{x_i}$$

Therefore,

$$E\left(\frac{x_i!}{(x_i-y_i)!}\right) = e^{-2L} \sum_{x_i=0}^{\infty} \frac{(1+L)^{x_i} L^{x_i}}{x_i!} = e^{-2L} \sum_{x_i=0}^{\infty} \frac{((1+L)L)^{x_i}}{x_i!}$$

and, by *Taylor's theorem*, which implies that $\sum_{x_i=1}^{\infty} \frac{((1+L)L)^{x_i}}{x_i!} = e^{L(L+1)}$ we have

$$E\left(\frac{x_i!}{(x_i-y_i)!}\right) = e^{-2L} e^{L^2+L} = e^{L^2-L} \quad (5.12)$$

Therefore, by substituting 5.12 in 5.11 we conclude that:

$$\sum_i E\left(\frac{x_i!}{(x_i-y_i)!}\right) = \sum_i e^{L^2-L} = de^{L^2-L} = LL_o e^{L^2-L}$$

Hence, formula 5.9 is valid.

By substituting 5.8 and 5.9 to 5.7, we obtain the following expression for the average complexity of the object distance algorithm:

$$T_x = (L_o L e^{L^2-L} + L_o^2 E_{map} + 2L_o E_{sal} + E_{in} + E_{isa}) \frac{(L_o L + 1)^{x-1}}{(L_o L + 1) - 1} + (E_{in} + E_{isa})(L_o L + 1)^x \quad (5.13)$$

Thus, the analysis of similarity between two objects considering only their immediate attributes and not traversing the entire transitive closures of their attributions, has an average time complexity, equal to:

$$T_1 = L_o L e^{L^2-L} + L_o^2 E_{map} + 2L_o E_{sal} + (E_{in} + E_{isa})(L_o L + 3) \quad (5.14)$$

Also, the average time complexity of similarity analysis considering attributes of two successive attribution levels is given by:

$$T_2 = (L_o L e^{L^2-L} + L_o^2 E_{map} + 2L_o E_{sal} + E_{in} + E_{isa})(L_o L + 2) + (E_{in} + E_{isa})(L_o L + 1)^2 \quad (5.15)$$

Formula 5.13 indicates that the main factor of the average computational cost of similarity analysis is the exponential e^{L^2-L} . It is important that this factor depends only on the average number of attributes in each of the semantically homogeneous categories (i.e. L) and not on the number of the attributes of two objects. Thus, even when the objects under comparison have many attributes, the average computational cost of similarity analysis can be reasonable, if L is low. L will be low when the attributes of objects are partitioned into many disjoint semantically homogeneous categories. It can be reasonably expected, that this will be the case, when conceptual models include a meta schema of attribute

classes, which introduces fine grain semantic distinctions between the relations in the application domain.

5.4 Performance Improvements

The non polynomial cost of computing the attribution distance in our current implementation has motivated an improvement.

This is a safe search heuristic in the algorithm for generating and selecting the optimal isomorphism between the attributes of two objects. As described in section 5.2.9, a partially developed isomorphism is pruned whenever its current distance (i.e. the sum of the distances between the attributes already mapped under it) exceeds the upper distance bound of the current optimal isomorphism.

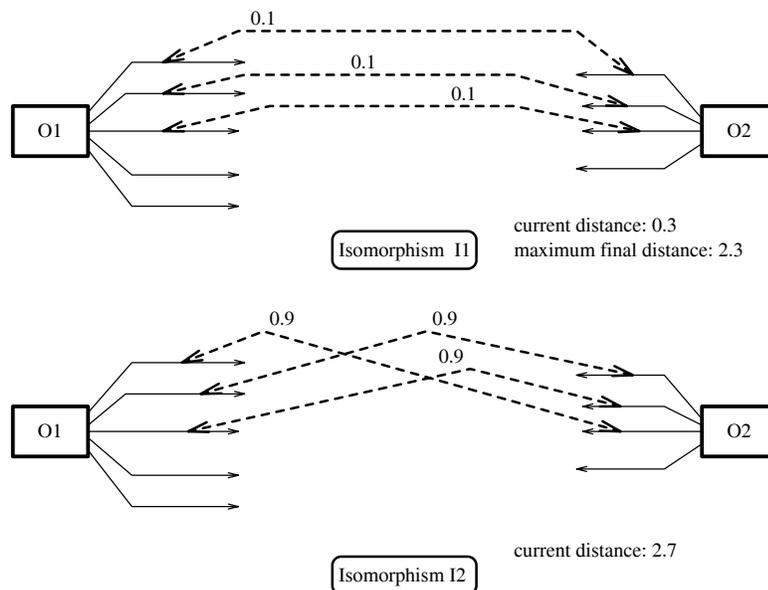


Figure 5.17: Pruning of Partial Isomorphisms

This is illustrated in figure 5.17. Assume that in selecting the optimal among all the possible isomorphisms between the attributes of objects O1 and O2, only the partial isomorphisms of the first three attributes of object O1 have been generated. Also assume that isomorphism I1 is the optimal between these partial isomorphisms. The current distance of I1 (i.e. the sum of the distances of the pairs of the mapped attributes that constitute it)

equals 0.3. Inevitably, one of the remaining attributes of O1 will be mapped onto no attribute of O2. This will introduce an additional distance of s^2 , where s is the salience of the unmapped attribute. Also, the distance between the remaining pair of attributes will be at most 1. Thus, regardless of the mapping of the remaining attributes of O1 onto the yet not mapped attributes of O2, this distance will never be greater than $1.3 + s^2$, which is always less than 2.3, since $s \leq 1$.

Notice that the current distance of I2 equals 2.7. Thus, further investigation into the possible completions of I2 is not necessary, since none of them can eventually have a distance lower than 2.7, and thus their overall distances will exceed the minimum upper distance bound of the currently optimal isomorphism I1.

The cost of similarity analysis can also be reduced in two additional ways.

First, we can reduce the number of evaluations of object distances by introducing look-up tables.

Figure 5.18 exemplifies such cases. The estimation of the distance between the objects O1 and O2 in this figure, requires the estimation of the distance between the attribute pairs (x,y), (z,e) and (k,h). However, the estimation of the distances between the first and the third of these pairs requires the estimation of the distance between the objects O3 and O4, twice.

To deal with such cases, we can maintain a table with distances of object-pairs, which have been evaluated in some similarity analysis scope. Thus, instead of reevaluating the distances between object-pairs, which have been already evaluated, we can retrieve them from the relevant table. This introduces the cost of searching the table with the evaluated distances but may lead to higher cost reductions. Analytic estimates of such reductions cannot be easily obtained. In general, they will depend on the frequency of reevaluations. This frequency depends on the number of the distinct pairs of semantically homogeneous attributes in the attribution graphs of objects, which also have identical object values.

Second, we can estimate the salience of attributes in a batch-mode and retrieve the relevant measures whenever they are required. The definitions of the belief functions that determine the salience estimates and the specification of the relevant algorithms indicate that salience estimates change in the following cases:

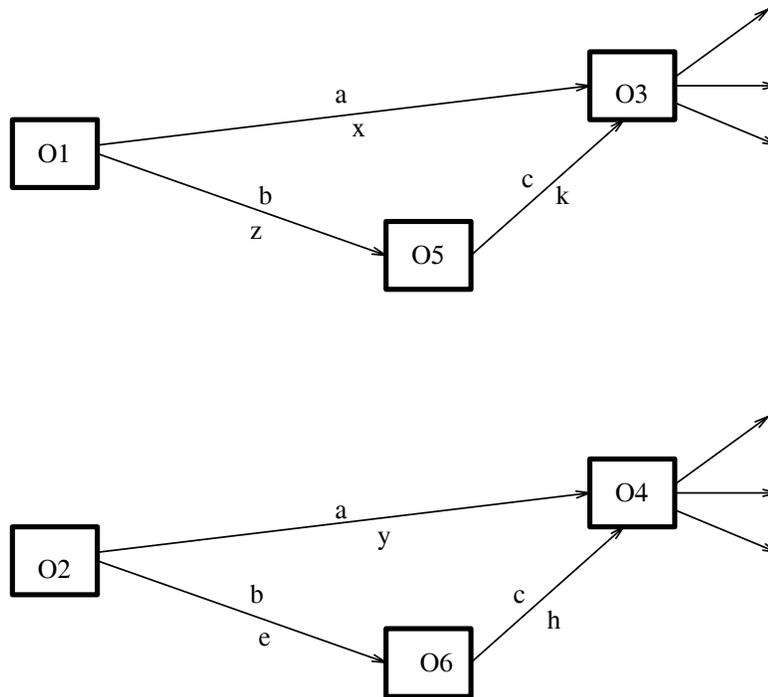


Figure 5.18: Reevaluation of Object Distances in Attribution Graphs

- when the classification of an attribute is modified;
- when the modeling of the classes of an attribute over the schema changes; and,
- when the extensions and/or the intensions of the classes in the scopes of the classes of an attribute are modified.

Thus, the salience of a particular attribute need only be reestimated in cases of conceptual model changes that belong to any of the previous cases. The estimation of salience in a batch mode could be implemented with an adequate triggering mechanism and would result into a maximum cost reduction equal to $\sum_{x_i} K_{sal} + \sum_{y_i} K_{sal}$ (which, on the average, equals $2L_o K_{sal}$), for the similarity analysis algorithm. In fact, such an implementation decision, could only be justified by an empirical study of the relative frequencies of the invocation of the *ObjectDistance* algorithm and the updates of the conceptual models.

5.5 The Prototype

The algorithms described in the preceding sections have been implemented in C++[Str87] and integrated with the *Semantic Index System(SIS)* [CD93] an outgrowth of the *Software Information Base*, developed in the Esprit project ITHACA[CDV93], as described in [Spa94a].

The Semantic Index System is a tool for describing large evolving varieties of concepts and complex relations, well suited for the representation of scientific knowledge and/or engineering designs. So far exemplar applications of the tool include[CD93]:

- the representation, documentation, analysis and retrieval of static properties of software source code, written in various programming languages; and,
- the construction of knowledge bases about descriptive, contextual and historical aspects of museum artifacts[CCD92].

The Semantic Index System offers a high performance knowledge management subsystem. It is capable of storing large populations of objects(10^6 is the latest tested order of magnitude) and querying them without significant performance degradations depending on the size of the stored population. It also offers multimode retrieval mechanisms(e.g. browsing, menu-based querying, hypertext facilities) for accessing stored objects and other database management facilities such as *transactions* and *concurrent multi-user access*.

These characteristics along with other operational features make SIS a fully operational platform, adequate for implementing and testing the similarity model. Furthermore, we intend to use similarity analysis in tasks of design and conceptual modeling by analogy in application domains such as software engineering and SIS has been acknowledged as a system adequate for representing design artifacts and conceptual models in such domains[CDP93,CDV93].

A prototype implementation of the similarity model, named Similarity Analyzer(SA) has been developed and integrated with the SIS. The SIS/SA system runs under UNIX system V and SUNOS and requires the X11 windows system(Rel 4 or 5) and the OSF Motif user interface library (Rel 1.1).

5.5.1 The Architecture of the SIS/SA

Figure 5.19 presents the architecture of SIS including SA. The module referred to as *SimilarityAnalyzer* comprises the implementations of the algorithms described in the preceding sections. This module is internally structured according to the diagram of figure 5.1.

In addition to the basic similarity analysis between objects, the *SimilarityAnalyzer* implements two further similarity-based queries. The first of them sorts the instances in the extension of a class in descending similarity order with respect to a specific object. The second sorts the instances in the extension of a class in descending *prototypicality* order. The prototypicality of objects in the extension of a class, is estimated as their average similarity with all the other instances of this class, according to the definition of prototypicality given by Tversky[Tve77].

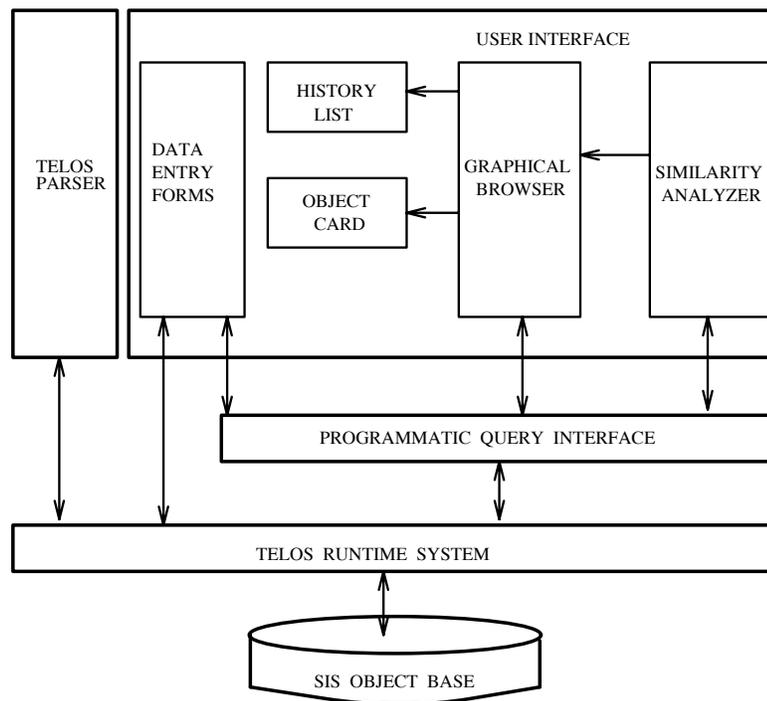


Figure 5.19: The General Architecture of SIS/SA

Similarities are computed between objects, stored in the *SIS Object Base*. SIS objects are described according to a structurally object-oriented data model, which descends from the Telos knowledge representation language [KMSB89, MBJK90]. This model is discussed in more detail in [CDV93].

In summary, it allows the structuring of knowledge as objects, uniquely identified by system identifiers, organized in an unbounded multiple classification hierarchy, with multiple strict generalization/specialization relations and having attributes. Attributes are object themselves. Thus, they may have their own attributes and also be organized through classification and/or generalization/specialization relations. Since objects in this data model are described according to the semantic modeling abstractions that are the foundation of our similarity model, their similarities can be analyzed directly from their internal SIS descriptions, without any representation transformations.

SIS objects can be specified using the syntax of the Telos language and then loaded into the SIS object base using the parser of the Telos language (i.e. the *TelosParser* module in figure 5.19). Examples of such specifications are given in appendix of this thesis, while a BNF grammar of the Telos language may be found in [KMSB89]. Alternatively, objects may be interactively specified and loaded into existing SIS object bases, using the *DataEntryForms* module, which supports the addition and/or modification of objects in an existing SIS object base, guided by the schema of this base.

As shown in figure 5.19, SIS objects are accessed by *SimilarityAnalyzer* through the *ProgrammaticQueryInterface* module, which provides a library of query-functions (see section 5.2.1 and [DD92] for a detailed description). These functions can be used for retrieving all the semantic relations involving a particular object, in a physically transparent way (i.e. the client of the programmatic query interface does not need to know the details of the internal representation of these relations in the object base).

For instance, the following query functions:

```

int get_all_classes(int set_id)
int get_all_superclasses(int set_id)
int get_inher_link(int set_id)

```

accept as input the identifier of a set containing the internal identifiers of one or more objects and return the identifiers of sets, which include the system identifiers of all the

classes, the superclasses and the attributes of all the objects identified by the system identifiers in the input set, respectively.

Furthermore, the *ProgrammaticQueryInterface* offers functions which manipulate the query answer sets (e.g. functions for taking the union and the difference of such sets or functions for inserting, deleting or searching elements in these sets).

SimilarityAnalyzer also communicates with the *GraphicalBrowser* module of the SIS, in order to display the results of similarity analysis (e.g. graphs representing the optimal isomorphisms between attributes of objects) in graphical forms (see figure 5.20 below).

5.5.2 Functional Features of the SIS/SA

The integration of *SimilarityAnalyzer* with SIS, essentially augments the functionality of this system with a basic mechanism for detecting analogies between objects.

In addition to other functional abilities, the user of the SIS/SA can:

- specify arbitrary conceptual models for various application domains, using the semantic modeling abstractions of classification, generalization and attribution;
- associate objects in such conceptual models with domain specific terms, whose natural meanings enable subsequent logical references to them (i.e. *logical names*);
- locate objects in conceptual models, using the following search mechanisms:
 - (i) browsing
 - (ii) menu-based querying
 - (iii) analogical similarity-based retrieval and,
 - (iv) primitive hypertext navigation ;

and

- detect and view - in graphical forms - analogies between objects.

As a descendant of SIS, the SIS/SA also inherits its other functional characteristics, such as the ability to reference multimedia objects, to customize the user interface of the tool and to write reports based on the answers of specific queries, automatically.

However, the isolation of the previous four functional features demonstrates the potential of SIS/SA as a platform for experimenting with applied forms of analogical reasoning,

such as *design by analogy*[Mos89] and incremental development of conceptual models by analogy. In the next chapter, we present illustrative applications of the SIS/SA in such tasks, which demonstrate the feasibility of using the similarity model to deal with them.

5.6 Experimental Evaluation

The SIS/SA system has been used in preliminary experiments of the similarity model. Our experiments concerned three basic behavioral aspects of the model:

- (i). The consistency of the similarity estimates computed by the model with similarity assessments provided by humans. Consistency is a prerequisite for employing the model in tools supporting analogical reasoning in cooperation with humans.
- (ii). The ability of similarity analysis to retrieve relevant analogs from a collection of sources, which is a necessary condition for successfully completing a session of analogical reasoning.
- (iii). The time performance of our model, which is critical for its practical utilization.

It is necessary to point out in advance that our experiments, although indicating a promising behavior of the similarity model, do not have a size that could safely support the general validity of the recorded results. Thus, they should be rather viewed as a preliminary experimental effort. This effort is bounded by both the sizes of the employed object bases and the diversity of their application domains. This limitation stems primarily from the unavailability of adequately large collections of information for testing the model in the context of real industrial tasks and environments. Thus, we had to resort to the current practice of experimentation with computational models of analogical reasoning, using limited sets of examples to demonstrate validity[FFG90,THNG90]. Additional information about the primitive material used in our experiments is presented in [Spa94b].

5.6.1 Ordering Consistency

The Purpose. The first of our experiments was carried out to evaluate the *consistency* of the estimates computed by similarity analysis as defined in this thesis, with similarity assessments provided by humans. The term consistency is not used in a strict logical sense but, rather to designate, how close the result of similarity analysis is to human intuition as expressed by similarity assessments.

Similarity analysis violating human intuition would inevitably have low pragmatic utility in cooperative systems of analogical reasoning(i.e. systems in which analogical conjectures about target analogs can only be drawn and verified by humans).

The Methodology. The selected application domain of our first experiment was that of computer programming. This domain was selected because of the availability of both human experts, that could provide similarity assessments and conceptual models with practical utilizations.

In particular, we used as experimental material a conceptual model of the C++ programming language[Str87]. This conceptual model specifies concepts of C++(e.g. *classes*, *operators* and *member functions*), in Telos. It also specifies the normal file structure of C++ programs, including dependencies between *source code files* and *header files*. Finally, it expresses dependencies between items in C++ programs such as function calls and accesses of variables. This model has been developed to support the automatic acquisition and storage of information about static properties of C++ programs[DK93].

In total, 10 cases of similarity analysis were carried out. In each case, a particular C++ concept(i.e. the *target concept*) was compared with a set of other C++ concepts(i.e. the set of *source concepts*). The elements of the source set were ranked in descending similarity order with respect to the target concept. The similarities between the compared concepts in each of these cases, were measured from the descriptions of the underlying C++ concepts in the employed conceptual model. Figure 5.20 presents an example of a comparison between the conceptual descriptions of a C++ class and a C++ structure.

The same 10 cases of a target and a set of source C++ concepts were given to 5 software engineers, ignorant of the results of the computational similarity analysis. All the subjects had at least two years of programming experience with C++. One of them had more than ten years of experience in industrial software engineering and held a Ph.D degree(subject 1 in tables 5.1 and 5.2). He also had a thorough knowledge of the conceptual model of C++. Another subject had a Bachelor's degree in Computer Science and two years of working experience(subject 2 in tables 5.1 and 5.2). The third subject (subject 3 in tables 5.1 and 5.2) had a Master's degree in Computer Science and the other two (subject 4 in table 5.1 and subject 5 in tables 5.1 and 5.2) were M.Sc students.

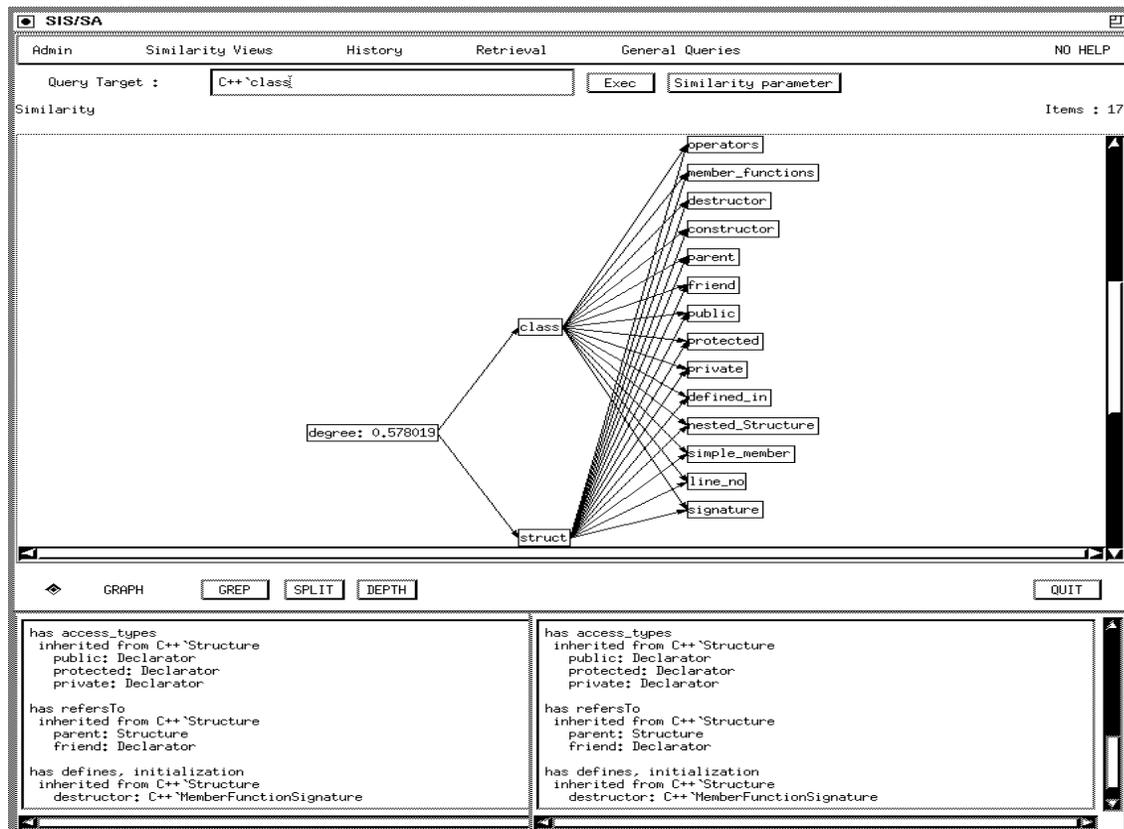


Figure 5.20: Similarity Between C++ Classes and Structures

The subjects were asked to rank the concepts in the source set with respect to their similarities to the target. The source item that would be assessed as the most similar to the target should be ranked first, the next most similar second and so on. Rank ordering of concepts based on similarity was preferred to numerical similarity estimates since as reported elsewhere[Sjo72,Tve77], sometime humans have difficulty in providing absolute similarity estimates.

The consistency between the ordering deduced from the similarity estimates computed by our model and the orderings produced by the subjects, was assessed in terms of the *rank correlation* between the relevant series. The measurement of the rank correlation between the orderings of the values of two variables is preferred from the measurement of the direct correlation of their values in cases where these values are not fully reliable[Kev84]. In our experiment, since the quality of the similarity estimates of the model was to be confirmed and the human assessments might be inaccurate, the rank correlation was the only reasonable choice.

In particular, the rank correlation between the ordering pairs, was measured by the Spearman coefficient r_s [Kev84, GW61]:

$$r_s = 1 - \frac{6 \sum_{i=1}^n (x_i - y_i)^2}{n(n^2 - 1)}$$

where x_i and y_i are the orders of corresponding values of the two variables, whose rank correlation is being estimated.

The Results. Table 5.1 presents the results of the rank correlation coefficients, that were measured between the orderings generated by the similarity analysis and those given by the human subjects. The first column of the table indicates the experimental case. The second column presents the size of the source set in each case and the other five present the measured correlation coefficients concerning the orderings provided by different subjects.

Case	Set Size	Subject 1	Subject 2	Subject 3	Subject 4	Subject 5
1	17	0.37****	0.5 **	0.47 ***	0.37 ****	0.46 ***
2	37	0.84 *	0.75 *	0.57 *	0.60 *	-0.06
3	18	-0.32	-	0.5 **	-0.13	0.12
4	18	-0.12	-	0.5 **	-0.18	0.10
5	4	0.55	0.55	0.55	0.55	0.55
6	17	0.55 **	0.60 *	0.55 **	0.51 **	0.64 *
7	4	0.55	0.85	0.65	0.55	0.4
8	18	0.57 *	0.48 **	0.87 *	0.65 *	0.52 **
9	17	0.42 ***	0.71 *	0.39 ****	0.50 **	0.5 **
10	19	0.52 **	0.87 *	0.73 *	0.86 *	0.64 *

Those coefficients marked by single(*), double(**), triple(***) and quadruple (****) asterisks were found statistically significant at the levels 0.01, 0.025, 0.05 and 0.1 respectively. The unmarked coefficients were not statistically significant. The tests for statistical significance in all but the second case were based on the theoretical distribution of the parameter D of the r_s coefficient ($D = \sum_{i=1}^n (x_i - y_i)^2$) [WG61]. The second case due to its large size was tested according to the *t-criterion* [Kev84].

Discussion. Evidently from table 5.1, in most cases, a positive, and statistically significant rank correlation coefficient between the orderings was found. It must also be pointed out that none of the negative coefficients was found statistically significant according to the tests used. All the subjects provided orderings leading to about equal numbers of positive rank correlation coefficients (i.e. 8, 8, 10, 8 and 9). Small differences were observed in the frequency of the statistically significant coefficients (i.e. 6, 6, 8, 6 and 5). Most of the measured coefficients indicate a moderate positive correlation (i.e. close to 0.5). Also, there were cases with a strong positive correlation (i.e. those where the r_s coefficient took values higher than 0.8, close to its upper bound which is 1). On the other hand, in all cases resulting in a negative correlation, the magnitude of the relevant coefficients was not large (i.e. none of them was less than -0.35, while the minimum possible value of r_s is -1.0).

In summary, this preliminary experiment provided evidence that similarity analysis generates measures, which do not violate human intuitions behind the assessment of similarity. However, this initial evidence needs to be verified by further experiments, which should investigate more thoroughly the dependency of consistent or inconsistent orderings on control variables such as the general expertise of the involved humans and their familiarity with the particular application domain.

5.6.2 Recall Evaluation

The Purpose. Using exactly the same conceptual model of C++ and four of the subjects who participated in the first experiment, we also measured the *recall* performance of similarity analysis.

Recall measures are traditionally used in evaluating the ability of information retrieval systems to locate as many as possible of the items in a collection, which are relevant to a given request [Su92]. Thus, recall reflects the completeness of answers. Completeness is important for analogical retrieval, since it reduces the possibility of ignoring relevant analogs in an analogical reasoning session. However, it does not necessarily correlate with the eventual success of analogical reasoning tasks. Other aspects, such as the comprehension of analogs and the validation of knowledge conjectures about target analogs, undoubtedly affect this success, too. Thus, we report our findings in order to conform with the evaluation practices for information retrieval systems, yet we are aware of

the limitations of such measures for evaluating systems for analogical retrieval, and especially systems for elaborating analogies, as the SimilarityAnalyzer.

The Methodology. Our second experiment was performed using the same experimental material for similarity analysis(i.e. the conceptual model of C++), that was utilized in the previous experiment and four of the five subjects that participated in it(subjects 1,2,3 and 5).

The subjects were given the same 10 cases consisting of a single target C++ concept and a set of source C++ concepts and were asked to indicate whether each of the source concepts could be used instead of the target concept in any programming task they could think of. Their answers, interpreted as indicating the analogy between the relevant concepts in programming, were used as judgements of relevance. On the basis of these judgements, we measured the recall of the sorted, in descending similarity order, source sets, using the formula

$$r = \frac{a_c}{R}$$

In this formula, a_c was the number of the relevant sources in the first c % positions of the sorted source-set and R the total number of the relevant items in the entire answer set. The use of *item cutoffs* (or *document cutoffs*) is necessary for measuring the recall performance of inexact retrieval mechanisms[Keen92]. Our choice to use a *proportional* instead of an *absolute* cutoff, was due to the need to attempt comparisons between the recall measures for source-sets with different cardinalities.

The Results. Table 5.2 presents the average recall measures of the 10 cases of analysis for each of the four subjects as well as the general average for all the subjects.

Cutoff	Subject 1	Subject 2	Subject 3	Subject 5	General
10%	0.33	0.35	0.34	0.44	0.35
25%	0.54	0.40	0.74	1.00	0.60
50%	0.83	0.80	0.91	1.00	0.86

The recording of the recall measures for the different subjects was necessary because their relevance judgements did not coincide for any of the experimental cases. Averages

were measured for three different cutoffs: the first 10%, 25% and 50% of the items in the sorted source sets.

Discussion. It is not possible to use these recall measures to compare our model, with other computational models of analogical retrieval, since such performance evaluations have not been reported in the literature to our knowledge. However, we believe that the recorded recall measures are encouraging since the first two of the selected cutoffs were fairly low. Also, the proportion of the items in any of the source sets, judged as relevant by any of the subjects did not exceed 25%. In fact, in the average only 10% of the items in these sets were assessed as being relevant to the target item.

5.6.3 Performance

The time performance of SIS/SA was measured in a variety of conceptual models and cases, differing in both the density and the way of utilizing the semantic modeling abstractions underlying similarity analysis.

Sessions of Type 1			Sessions of Type 2		
Attributes	RL_1	RL_n	Attributes	RL_1	RL_n
3.23	0.276	0.392			
3.7	0.43	0.589			
5.12	0.516	1.529	5.38	0.285	0.358
9.8	0.831	2.262			
13.94	0.926	2.522			
16.68	1.541	2.642	17.65	0.69	0.719
26.66	4.6	9.4			

Table 5.3 presents times required for similarity analysis between pairs of objects, as they were averaged over iterative analysis sessions.

All the times in table 5.3, which were measured for analysis sessions performed on Sun sparstations, are given in seconds. The size of each session is expressed as the average number of attributes per object.

Table 5.3 presents two time measures for each session. The measures of the first type, which are presented by the columns RL_1 , express the time spent for performing similarity analysis up to one level of recursion. Under this type of recursion control, the analysis did not exhaustively search the entire transitive closures of the attribution graphs of the objects. Instead, it estimated the pairwise distances between the attributes of the object/values of the attributes of the initial objects only from their classification and generalization/specialization relations(see the definition of the C factor in formula 5.5. The time measures, presented by the columns RL_n refer to exhaustive analysis of all the attributes in the transitive closures of the attribution graphs of the involved objects.

The distinction between these two levels of recursion, revealed the possible gains in time performance. It must be pointed out, that according to our experience, from the conceptual models we have experimented with so far, similarity estimates obtained from analysis at depths of 1 or 2 in attribution graphs of objects, do not differ significantly from the estimates obtained from analysis with uncontrolled recursion. In fact, this is the result of the definition of the overall distance D between attribute objects, which realizes as dominant factor the generalization distance between two attributes and not the distance between their object/values(see definition 3.34 in chapter 3).

The recorded times indicate a time performance of the similarity analysis model, which is comparable to the time performance of other computational models of analogical reasoning reported in the literature[FFG90].

In the same table we also distinguish between two types of analysis sessions. The first of these types(i.e. sessions of type 1 in the table) refer to similarity analysis of objects with only a few or no attributes inherited from the same ancestor(i.e. attributes which share the same original attribute class). The second type refers to similarity analysis of objects with a moderate or high number of attributes inherited from the same ancestor.

The recorded times for these two types of sessions differ substantially, regarding analysis of objects of about the same size and indicate a speed up of similarity analysis that may achieved in cases of conceptual models extensively utilizing the inheritance of attributes along interobject generalization relations. This is due to the definition of the partial distance metric d_4 , which by theorem 3.10, is minimized by isomorphisms, that necessarily map the attributes with common original classes(i.e. attributes inherited and possibly

refined from the same ancestor).

5.7 Summary

This chapter presented the algorithms for computing the distance and the salience measuring functions of the similarity model. It also investigated the time and the space complexities of these algorithms and described a prototype implementation.

This implementation was used in a set of experiments, aiming at

- demonstrating the consistency of the similarities computed by our model with assessments of similarities provided by humans;
- evaluating the recall performance of similarity analysis as a mechanism of analogical retrieval; and
- obtaining evidence about the time performance of the model.

Albeit preliminary, the findings of these experiments indicated that similarity analysis generates estimates consistent with human assessments of similarity and furthermore has a good recall performance as a model of analogical retrieval.

The basic limitation of our current implementation is its exponential worst case time complexity. However this complexity could be significantly reduced, as discussed in section 5.3.1. Furthermore, in various cases similarity analysis can be performed in reasonable time depending on the very modeling of the involved objects. As derived analytically and verified empirically, similarity analysis is not overly computationally expensive whenever objects are described according to meta models making fine grain semantic distinctions between different kinds of properties and relations in various application domains and utilizing the generalization abstraction for enabling inheritance of common attributes from superclasses to subclasses of objects.

In summary, our findings indicate an encouraging behavior of the model and make worth further evaluation in practical tasks, on undertaking, however, beyond the scope of this thesis.

Chapter 6

Illustrative Applications

6.1 Introduction

In this chapter, we present illustrative applications of similarity analysis in requirements engineering. In particular, we show how it may be used in :

- modeling and eliciting requirements specifications for software systems by reusing analogous existing specifications; and
- analyzing different specifications of the same system in order to integrate them into a single one.

The selection of the application area of requirements engineering and the particular tasks in it, has been motivated by several reasons.

- Requirements engineering is itself a crucial stage of software development. Accurate and complete specifications reduce the probability of subsequent wrong decisions, in system development. Thus, successful requirements engineering affects the entire system development cost[Boeh89].

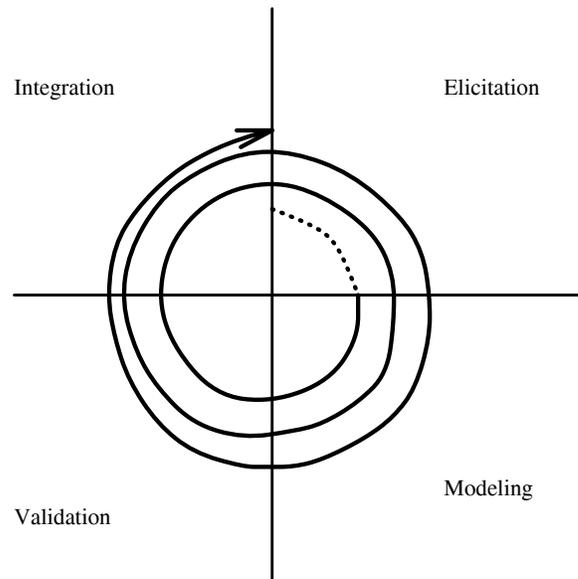


Figure 6.1: The Spiral Process of Requirements Engineering

- The selected tasks are important for delivering complete, consistent, unambiguous and agreed upon specifications[Hof93,Pohl93]. Thus, they are proposed as three of the four distinct stages of requirements engineering in models describing it as an activity[LF91,RW91,Hof93](see figure 6.1).

Notice that requirements specifications may be

- fairly complex, since they encompass functional and non functional aspects of a system and relate these aspects with elements in its operational context[MBJK90,RW91,Hof93].
- ambiguous, due to the informal communication between agents participating in the development of a system, who are experts either in the application domain of the system(i.e. clients) or in technical issues(i.e. software engineers).
- incomplete and inconsistent, especially during the inner cycles of the spiral requirements engineering process(see figure 6.1)
- generated according to different specification meta models, subject to the modeling practice and the tools available to the involved software engineers(a recent survey has

pointed out that 8 different specification languages are currently adopted by commercial upstream CASE tools[Fug93]).

- not generalizable into stable abstractions of fixed granularity, since the construction of such abstractions is still an open research issue (see [Maid92] for a similar argument), depending on the subsequent operations that these abstractions are meant to accomplish.

These characteristics devalidate a number of assumptions usually made by computational models of analogical reasoning. The representation of analogs cannot be assumed to use fixed relation and concept types. Also, the existence of particular kinds of abstractions supporting the detection of analogies (e.g. fixed vocabulary denoting representative concepts in some domain and predefined relations between them) is questionable. Thus, elaboration of analogies between specifications is a hard task.

6.2 Analogical Reuse of Requirements

6.2.1 Reuse of Specifications.

It was not until 1983, when Freeman[Fre83], proposed that reuse should not be limited to source code fragments and could include *upstream* software artifacts, such as specifications, designs, and documentations, that the research community started to consider the issues of reusing such higher level artifacts.

Successful reuse of existing requirements specifications leads to significant productivity and cost gains. This is because it reduces the possibility of coming up with incomplete and/or wrong specifications that may lead to later improper design and implementation decisions with substantial delays and adverse cost effects to the whole project. It is particularly useful in cases where it is not possible to specify requirements with extensive co-operation with the clients of the software system or when such clients have limited experience and knowledge about the application domain[Hof93](e.g. co-operation with new personnel). Reuse of specifications may also promote reuse of design and code artifacts, if specifications are directly associated with such artifacts[Tr90,CDV93].

However, reuse of specifications differs substantially from reuse of downstream artifacts.

First of all, it cannot be based on a black box strategy, where artifacts are reused without being modified subject to certain usability conditions (e.g. reuse from libraries of source code components). The complexity of existing specifications makes them very unlikely

to reflect all the aspects of new systems exactly[MBJK90,RW91,Hof93,Pohl93]. Even specifications of systems in single application domains, with identical technical concerns, may differ due to the distinct operational contexts of these systems. Thus, reuse of requirements must necessarily resolve all the operational problems of reuse[BR87]. In other words, reusable specifications must be selected from some repository, comprehended, adapted so that to precisely reflect the objectives of the new system and eventually integrated into a new specification. Thus, reuse of requirements is an instance of *compositional reuse*[Diaz93].

The selection, comprehension and adaptation of reusable specifications is particularly difficult. Selection for reuse is critically affected by the presence of abstractions that provide succinct descriptions suppressing all the irrelevant details of reusable components and retaining only those which are really important for reusing them[Kru92]. However, the formation of such abstractions is difficult in software engineering in general[Kru92] and even harder for upstream artifacts, such as requirements specifications. This is evident from the proposal of many different kinds of abstractions for requirements specifications(e.g. *domain abstractions* in [Maid92], *cliches* in [RW91], *abstract data flow schemas* in [LH86], *operation schemas* in[MH91]) and the absence of an agreement about the nature and the granularity of such abstractions, even in the research community, not to mention the industry.

The comprehensibility of specifications is hampered by the diversity of the terminology used in different application domains and organizational environments, which may not be always familiar to software engineers. It may also be restricted by the complexity of specifications, and their representation according to different specification metamodels[JFH92].

All these difficulties also affect the modifiability of retrieved specifications.

6.2.2 Analogical Reasoning as a Paradigm for Reuse

Analogical reasoning has been proposed as a paradigm for compositional reuse of specifications[MS91,MH91a,MH91b,MS92], which can offer appropriate solutions to its operational problems.

The mapping between the elements of source and target analogs, corresponding to existing and on-going specifications, can be utilized for deriving conjectures promoting reuse.

These conjectures are derived from three kinds of specification elements that can be distinguished with respect to an established analogical mapping, namely (see *figure 6.2*):

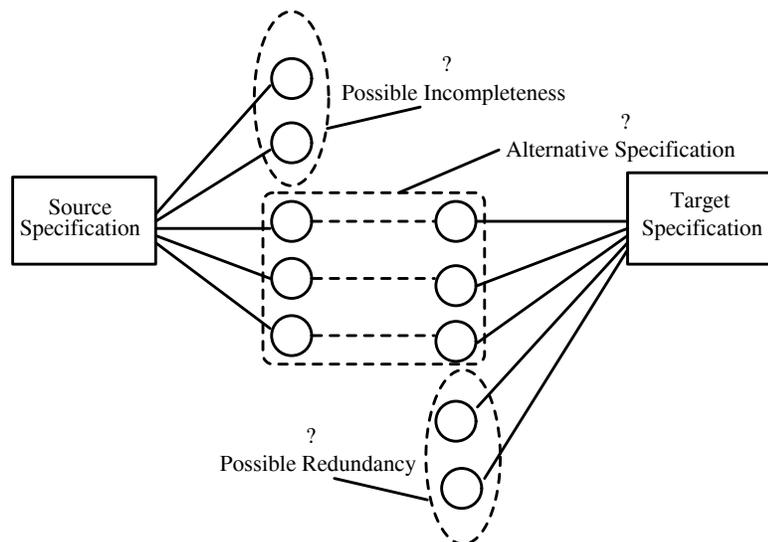


Figure 6.2: Pairwise Analogies Between Source and Target Specifications

- the corresponding elements of two specifications given an analogical mapping between them;
- the unique elements of the source (i.e. elements of the source not mapped onto any of the target elements); and,
- the unique elements of the target (i.e. elements of the target not mapped onto any of the source elements).

Elements of these kinds give rise to different types of conjectures about the target specification. First, the mapped elements of the target may be changed according to the modeling of their source counterparts. This conjecture can improve the accuracy of the target specification, with respect to the client needs, provided that the source specification reflects these needs. Second, the unique elements of the source specification may be missing from the target specification and consequently they could be imported to it, thus improving its completeness. Finally, the unique elements of the target may be redundant and consequently they could be removed from the target

specification.

In spite of the intuitive appeal of these conjectures, it must be emphasized that none of them is valid unless it is confirmed by the software engineer, who develops the target specification. This rather conservative approach, which is shared by some authors(see [Maid92]) and is neglected by others (see [LF91]) realizes that there is no generally applicable objective criterion for validating analogical inferences[Hall89].

Analogies between specifications, also act as catalysts to the comprehension of the source specification. This is because, an unfamiliar object(i.e. the source specification), can be understood more easily when exposed in terms of another familiar object(i.e. the target specification). In our case, the mapping of the target specification, which directly reflects the current understanding of a client's objectives by the software engineer, onto an unfamiliar source specification promotes the comprehension of the latter. This effect of analogies to comprehensibility has been empirically verified in[MS92,SM92,Maid92].

Furthermore, analogical reasoning is able to support reuse across different domains and thus it has the potential of exploiting specifications from a wide variety of applications[Maid92].

6.3 Similarity Based Analogical Reuse of Specifications

The following examples illustrate how similarity analysis detects analogies between specifications, how these analogies can be utilized in producing specifications by reuse and that similarity analysis can be applied to specifications of different domains, devised according to various specification models.

6.3.1 Example 1: Detection of Analogies between Specifications

The first example illustrates the detection of analogies between specifications through the analysis of similarity.

Figure 6.3 presents two data flow diagrams representing activities of software systems supporting the operations of a hypothetical library and a car renting agency. The first of these diagrams, models the necessary stages for logging in the computer system of the library in order to search for some book and possibly reserve it for borrowing. The second diagram models the stages for picking up a car model and renting a car of this model, using an automatic transaction machine(ATM) of the car renting agency.

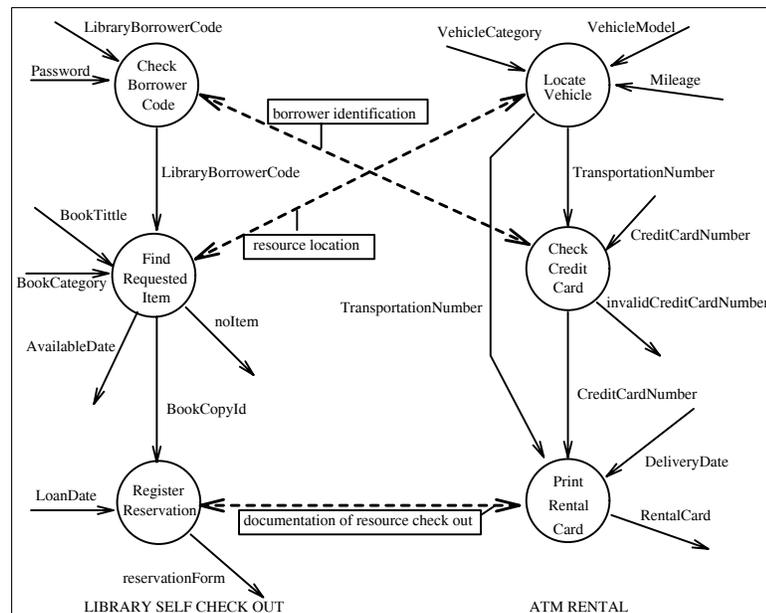


Figure 6.3: The Analogy of Distant/Electronic Borrowing of Resources

Assuming that activities and data flows of data flow diagrams correspond to transactions and their inputs and/or outputs, respectively, data flow diagrams can be described as instances of a meta model, which is defined in terms of the semantic modeling abstractions of our framework. Figure 6.4 presents a meta model for describing transactions, defined according to the formal notation introduced in the third chapter.

In these formal definitions, logical names act as unique object identifiers to improve their readability. The formal definitions of figure 6.4, are graphically illustrated by figure 6.5.

This meta model enables the description of *transactions* which accept as input and produce as output *entities*. Transactions are decomposed in other transactions. Activities of data flow diagrams can be defined as instances of the *TransactionClass* and then be associated with attributes, instantiating some of the three attribute classes defined for this class (i.e. the attribute classes *input*, *output* and *consistsOf*). Input and output attributes will express their input and output information flows, respectively. Decomposition attributes can be used to associate activities, with lower level data flow diagrams describing

$[TransactionClass, TransactionClass, nil, \{ \#C3, \#C, \#I \}, \{ \#C2 \}, \{ input, output, consistsOf \}, nil]$
 $[input, input, TransactionClass, \{ \#C3, \#C, \#A \}, \{ \#C2 \}, \emptyset, EntityClass]$
 $[output, output, TransactionClass, \{ \#C3, \#C, \#A \}, \{ \#C2 \}, \emptyset, EntityClass]$
 $[consistsOf, consistsOf, TransactionClass, \{ \#C3, \#C, \#A \}, \{ \#C2 \}, \emptyset, TransactionClass]$
 $[EntityClass, EntityClass, nil, \{ \#C3, \#C, \#I \}, \{ \#C2 \}, \emptyset, nil]$

Figure 6.4: Formal Definition of a Meta Model for Describing Transactions

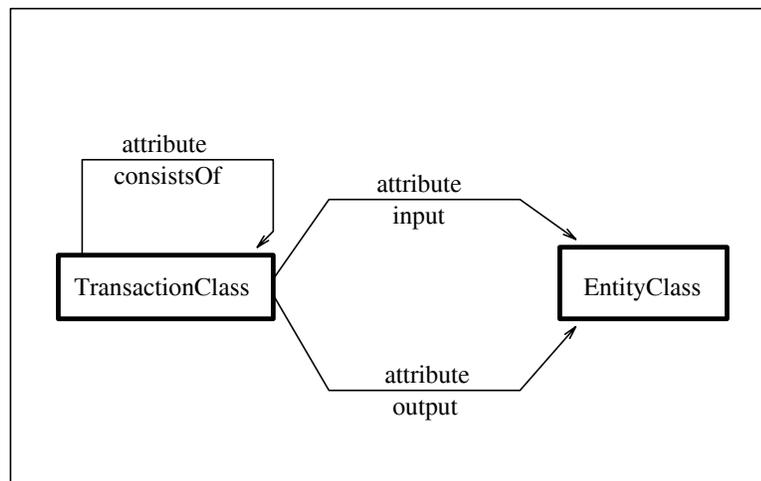


Figure 6.5: Graphical Form of the Meta Model for Describing Transactions

them in more detail. In this sense, zero-level data flow diagrams (also referred to as *context data flow diagrams*) correspond to one transaction, and detailed refinements of such diagrams, correspond to decompositions of this transaction.

For instance, the data flow diagrams of figure 6.3 are formally defined as instances of the *TransactionClass*, as presented in figures 6.6 and 6.7. Their definition as SIS/SA objects using the syntax of Telos language is presented in the appendix of this thesis.

$[LibrarySelfCheckOut, LibrarySelfCheckOut, nil, \{ \#C2, \#C, \#I, TransactionClass \}, \{ \#C1 \},$
 $\{ checkBorrowerCode, findRequestedItem, registerReservation, libraryBorrowerCode, password,$
 $bookTitle, bookCategory, loanDate, noItem, reservationForm, availableDate \}, nil]$

$[checkBorrowerCode, checkBorrowerCode, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, consistsOf \}, \{ \#C1 \},$
 $\emptyset, CheckBorrowerCode]$

$[findRequestedItem, findRequestedItem, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, consistsOf \}, \{ \#C1 \},$
 $\emptyset, FindRequestedItem]$

$[registerReservation, registerReservation, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, consistsOf \}, \{ \#C1 \},$
 $\emptyset, RegisterReservation]$

$[libraryBorrowerCode, libraryBorrowerCode, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, input \}, \{ \#C1 \},$
 $\emptyset, LibraryBorrowerCode]$

$[password, password, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, input \}, \{ \#C1 \}, \emptyset, Password]$

$[bookTitle, bookTitle, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, input \}, \{ \#C1 \}, \emptyset, BookTitle]$

$[bookCategory, bookCategory, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, input \}, \{ \#C1 \}, \emptyset, BookCategory]$

$[loanDate, loanDate, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, input \}, \{ \#C1 \}, \emptyset, Date]$

$[noItem, noItem, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, output \}, \{ \#C1 \}, \emptyset, NoItem]$

$[reservationForm, reservationForm, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, output \}, \{ \#C1 \},$
 $\emptyset, ReservationForm]$

$[availableDate, availableDate, LibrarySelfCheckOut, \{ \#C2, \#C, \#A, output \}, \{ \#C1 \}, \emptyset, Date]$

Figure 6.6: Formal Definition of the LibrarySelfCheckOut transaction

Both data flow diagrams aggregate and distinguish between attributes that express their inputs, outputs and decompositions. This distinction is possible by the classification of their attributes under only one of the attribute classes *input*, *output* and *consistsOf*.

$$\begin{aligned}
& [ATMVehicleRental, ATMVehicleRental, nil, \left\{ \#C2, \#C, \#I, TransactionClass \right\}, \left\{ \#C1 \right\}, \\
& \left\{ locateVehicle, checkCreditCard, printRentalCard, vehicleModel, vehicleModelCategory \right. \\
& \left. mileage, creditCardNumber, deliveryDate, rentalCard, invalidCreditCard \right\}, nil] \\
& [locateVehicle, locateVehicle, ATMVehicleRental, \left\{ \#C2, \#C, \#A, consistsOf \right\}, \left\{ \#C1 \right\}, \emptyset, LocateVehicle] \\
& [checkCreditCard, checkCreditCard, ATMVehicleRental, \left\{ \#C2, \#C, \#A, consistsOf \right\}, \left\{ \#C1 \right\}, \\
& \emptyset, CheckCreditCard] \\
& [printRentalCard, printRentalCard, ATMVehicleRental, \left\{ \#C2, \#C, \#A, consistsOf \right\}, \left\{ \#C1 \right\}, \emptyset, PrintRentalCard] \\
& [vehicleModel, vehicleModel, ATMVehicleRental, \left\{ \#C2, \#C, \#A, input \right\}, \left\{ \#C1 \right\}, \emptyset, VehicleModel] \\
& [vehicleModelCategory, vehicleModelCategory, ATMVehicleRental, \left\{ \#C2, \#C, \#A, input \right\}, \left\{ \#C1 \right\}, \\
& \emptyset, VehicleModelCategory] \\
& [mileage, mileage, ATMVehicleRental, \left\{ \#C2, \#C, \#A, input \right\}, \left\{ \#C1 \right\}, \emptyset, Mileage] \\
& [creditCardNumber, creditCardNumber, ATMVehicleRental, \left\{ \#C2, \#C, \#A, input \right\}, \left\{ \#C1 \right\}, \\
& \emptyset, CreditCardNumber] \\
& [deliveryDate, deliveryDate, ATMVehicleRental, \left\{ \#C2, \#C, \#A, input \right\}, \left\{ \#C1 \right\}, \emptyset, Date] \\
& [rentalCard, rentalCard, ATMVehicleRental, \left\{ \#C2, \#C, \#A, output \right\}, \left\{ \#C1 \right\}, \emptyset, RentalCard] \\
& [invalidCreditCard, invalidCreditCard, ATMVehicleRental, \left\{ \#C2, \#C, \#A, output \right\}, \left\{ \#C1 \right\}, \\
& \emptyset, InvalidCreditCardNumber]
\end{aligned}$$

Figure 6.7: Formal Definition of the ATMVehicleRental transaction

The Essence of the Analogy. The analogy between the transactions *LibrarySelfCheckOut* and *ATMVehicleRental* regards the basic stages of processing for borrowing a resource, through a distant interaction with a software system, including:

- i. the identification and validation of the person who wants to borrow some resource;
- ii. the location of this resource; and
- iii. the commitment of the borrowing activity.

The dashed lines in *figure 6.3* illustrate the analogies between these stages in the particular transactions of our example.

Notice that neither the exact sequence, nor the input or the output information produced by these stages are the same. For instance, the validation of a resource borrower by the library system is a prerequisite for allowing him to search for the required library item. On the other hand, a client can interact with an ATM machine of the car renting agency system without being identified, unless he has found a vehicle fitting his needs and he wants to rent it.

Also, identification in these two transactions is based on different, albeit analogous, sorts of information. The library borrower is identified through a password ensuring its legal and privileged access to the library software system and a code associated with him for general accounting purposes. On the other hand, the validation of a customer of the car renting agency is possible only through his credit card number, which also allows an immediate charging for the rent.

In spite of their differences, credit card numbers, library codes and passwords are analogous, as unique identifiers of borrowers. Similarly, transportation numbers and identifiers of book copies are analogous as unique identifiers of resources.

Detection of Analogies. The analysis of similarity reveals the outlined analogies between the involved transactions, through the estimation of their distance over attribution. Notice that, the particular transactions share no common superclass abstracting their semantic similarities. Furthermore, their classification under *TransactionClass* gives only a rough indication of their common substance.

The estimation of the attribution distance between the *LibrarySelfCheckOut* and the *ATMVehicleRental* is based on the evaluation of an isomorphism between their attributes, which minimizes the sum of the distances of the pairs of attributes that constitute it.

Due to the criterion of the semantic homogeneity, only attributes, which are both classified under either the attribute class *input*, or the attribute class *output* or the attribute class *consistsOf* can be mapped onto each other.

Let us focus on how an optimal isomorphism is determined between the decomposition attributes(i.e. the attributes classified under the attribute class *consistsOf*) of the involved

transactions. The three subtransactions of the *LibrarySelfCheckOut* can be mapped onto the three subtransactions of the *ATMVehicleRental*, according to six different isomorphisms shown in figure 6.8.

Isomorphism 1:

checkBorrowerCode \longleftrightarrow *locateVehicle*
findRequestedItem \longleftrightarrow *checkCreditCard*
registerReservation \longleftrightarrow *printRentalSlip*

Isomorphism 2:

checkBorrowerCode \longleftrightarrow *locateVehicle*
findRequestedItem \longleftrightarrow *printRentalSlip*
registerReservation \longleftrightarrow *checkCreditCard*

Isomorphism 3:

checkBorrowerCode \longleftrightarrow *checkCreditCard*
findRequestedItem \longleftrightarrow *locateVehicle*
registerReservation \longleftrightarrow *printRentalSlip*

Isomorphism 4:

checkBorrowerCode \longleftrightarrow *checkCreditCard*
findRequestedItem \longleftrightarrow *printRentalSlip*
registerReservation \longleftrightarrow *locateVehicle*

Isomorphism 5:

checkBorrowerCode \longleftrightarrow *printRentalSlip*
findRequestedItem \longleftrightarrow *checkCreditCard*
registerReservation \longleftrightarrow *locateVehicle*

Isomorphism 6:

checkBorrowerCode \longleftrightarrow *printRentalSlip*
findRequestedItem \longleftrightarrow *locateVehicle*
registerReservation \longleftrightarrow *checkCreditCard*

Figure 6.8: Isomorphisms between the LibrarySelfCheckOut and the ATMVehicleRental

The selection of the optimal isomorphism among them is based on their overall distances, which require the evaluation of the distances between all the values of the involved pairs of attributes and therefore the analysis of similarity between the following pairs of subtransactions:

CheckBorrowerCode , *LocateVehicle*

CheckBorrowerCode , *CheckCreditCard*

CheckBorrowerCode , *PrintRentalSlip*

FindRequestedItem , *LocateVehicle*

FindRequestedItem , *CheckCreditCard*

FindRequestedItem , *PrintRentalSlip*

RegisterReservation , *LocateVehicle*

RegisterReservation , *CheckCreditCard*

RegisterReservation , *PrintRentalSlip*

However, the estimation of the overall distances between these transaction-pairs, also depends on their attribution distances and therefore it makes necessary the evaluation of further optimal isomorphisms between their own attributes. This process will detect the analogies between the pairs of subtransactions (*CheckBorrowerCode,CheckCreditCard*), (*FindRequestedItem,LocateVehicle*) and (*RegisterReservation,PrintRentalSlip*) and thus, it will end up with the selection of the third from the isomorphisms of figure 6.8.

Let us investigate the analogy between the *FindRequestedItem* and the *LocateVehicle* subtransactions and show how this analogy is detected.

As illustrated in figure 6.9, these two transactions accept input information, which enables the location of a resource according to the needs of the potential borrower and output a unique identifier for a located resource(if any). Also, they involve analogous substages of processing(i.e. further decomposing subtransactions). The search for a resource is gradual since it proceeds through two basic stages.

The first of them determines some general category of resources, satisfying the needs of a borrower. This stage corresponds to the single subtransaction *DetermineBookCategory* of the transaction *FindRequestedItem* and to the subtransactions *SelectModelCategory* and *SelectVehicleModel* of the transaction *LocateVehicle*. The second general stage selects a specific resource within the category selected by the preceding stage. This stage corresponds to the subtransactions *SelectBookByTitle* and *SelectVehicle*.

Assuming that none of the subtransactions of the transactions *FindRequestedItem* and *LocateVehicle* is further decomposed, the evaluation of an optimal isomorphism between them, depends on the estimation of their distances with respect to their inputs and outputs. These distances are lower for the transaction-pairs *SelectBookByTitle, SelectVehicle* and *DetermineBookCategory, SelectModelCategory*. Therefore, these transactions will be mapped onto each other, eventually.

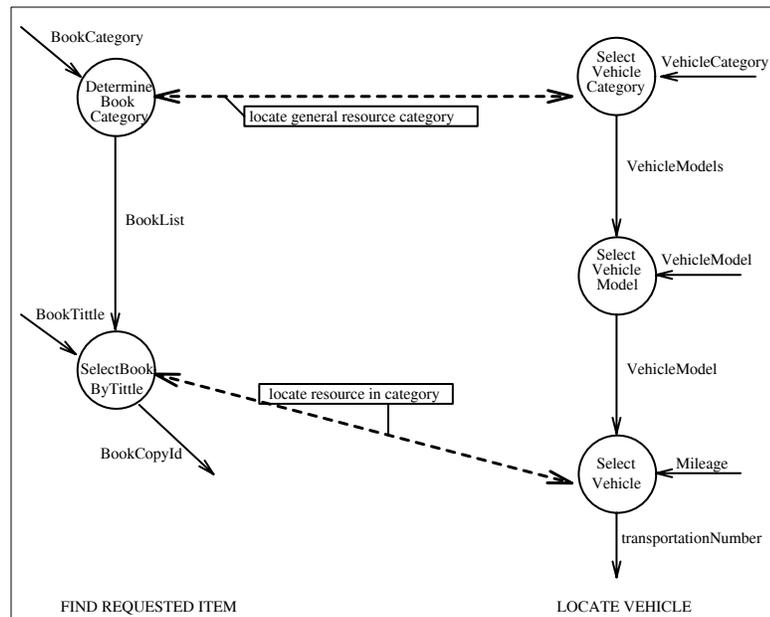


Figure 6.9: Analogies Between the FindRequestedItem and LocateVehicle Subtransactions

The lower distances between the inputs and the outputs of these transactions result from their modeling.

Let us assume that all input and output entities are classified only under the class *EntityClass* and that none of them has attributes of its own. Then their pairwise overall distances will differ only due to their generalization distances. Figure 6.10 presents their generalization graph.

According to this graph, the relevant interentity absolute generalization distances are estimated, as shown in table 6.1. From this table, it can be verified that the optimal isomorphism between the attributes of the subtransactions *SelectBookByTitle* and *SelectVehicle*, will include the pairs of attributes:

$$\text{input:bookTitle , mileage} \\ (D_3(\text{BookTitle}, \text{Mileage})=0.5) ,$$

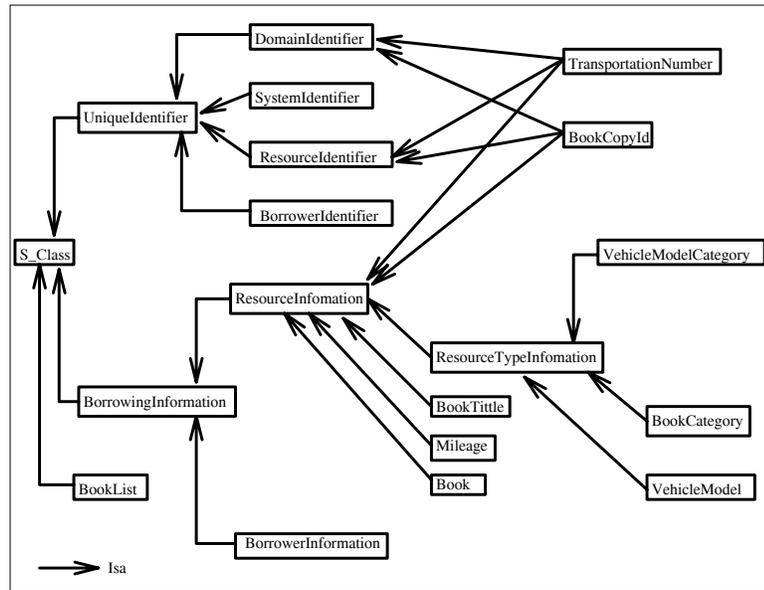


Figure 6.10: Generalization Graph of Input and Output Entities

Pair of Entity Classes	Distance
BookCopyId, TransportationNumber	0.4
BookCategory, VehicleModelCategory	0.4
BookCategory, VehicleModel	0.4
BookTitle, Mileage	0.5
BookTitle, VehicleModel	0.7
BookTitle, VehicleModelCategory	0.7
BookCategory, Mileage	0.7
BookCopyId, VehicleModel	0.95
BookList, Mileage	1.08
BookList, VehicleModel	1.28
BookList, TransportationNumber	1.28
BookList, VehicleModelCategory	1.28

input: *bookList, vehicleModel*
 $(D_3(\text{BookList}, \text{VehicleModel})=1.28)$ and,
output: *bookCopyId, transportationNumber*
 $(D_3(\text{BookCopyId}, \text{TransportationNumber})=0.4)$

The overall distance of this optimal isomorphism between the subtransactions *SelectBookByTitle* and *SelectVehicle* is lower than the overall distance of the optimal isomorphism between the transactions *SelectBookByTitle* and *SelectVehicleModel*, which consists of the following pairs:

input: *bookTitle , vehicleModel*
 $(D_3(\text{BookTitle}, \text{VehicleModel})=0.7)$

input: *bookList , selectedModels*
 $(D_3(\text{BookList}, \text{VehicleModel})=1.28)$

output: *bookCopyId , selectedModel*
 $(D_3(\text{BookCopyId}, \text{VehicleModel})=0.95)$

Also, it is lower than the overall distance of the optimal isomorphism between the transactions *SelectBookByTitle* and *SelectModelCategory*, which consists of the following pairs:

input: *bookTitle , vehicleModelCategory*
 $(D_3(\text{BookTitle}, \text{VehicleModelCategory})=0.7)$

input: *bookList , nil*
(it has the maximum absolute interattribute distance, which is ∞)

output: *bookCopyId , selectedModel*
 $(D_3(\text{BookCopyId}, \text{VehicleModel})=0.95)$

Thus, the best candidate for mapping the transaction *SelectBookByTitle* is the transaction *SelectVehicle*. Similarly, it can be shown that, the aggregate distance of the optimal isomorphism between the attributes of the transaction *DetermineBookCategory* and the attributes of the transaction *SelectModelCategory* is lower than the aggregate distance of the corresponding isomorphism between the former transaction and the transaction *SelectVehicleModel*. Hence, *DetermineBookCategory* will be mapped onto the transaction *SelectModelCategory*.

Discussion. Certain observations about the analysis of similarity, in reference to the first example, make evident that its ability to detect analogies is rather tolerant to different degrees of utilization of the classification and the generalization modeling abstractions.

In fact, apart from the very rough partition of the objects into *transactions* and *entities* and the attributes into *input*, *output* and *decomposing* ones, no other classification relations were taken into account in the preceding analysis. It must also be pointed out that similarity analysis would result into the same isomorphisms between the transactions *LibrarySelfCheckOut*, *ATMVehicleRental*, and the subtransactions *FindRequestedItem*, *LocateVehicle* even if their attributes were not partitioned into the prescribed categories (i.e. input, output and consistsOf).

In this hypothetical case, the criterion of semantic homogeneity, would allow comparisons between all possible pairs of attributes of the involved transactions. The analysis would be more expensive because it could not preclude comparisons between semantically heterogeneous attributes, such as input and output ones, in advance. However, the eventual semantic resemblances between the inputs and the outputs of the subtransactions of the *LibrarySelfCheckOut* and *ATMVehicleRental* would lead to the selection of the same isomorphisms.

Also, the generalization relationships, that were taken into account, involved only the input and the output entities of the compared transactions. However, as it was evident from the eventual isomorphisms, there were semantic resemblances between some of these transactions, that could have been expressed by further generalizations of the relevant specifications.

Figure 6.11 presents possible such generalizations involving the specifications of the *FindRequestedItem* and the *LocateVehicle* transactions. These two transactions are generalized into a common general transaction, called *SearchResource*, which abstracts their analogous stages of processing (i.e. the location of a resource category and the location of a resource). These stages are represented by the decomposing attributes *locateResource* and *locateResourceCategory*, which have as values two abstract subtransactions, namely the subtransactions *LocateResourceCategory* and *LocateResource*. These attributes generalize the pairs of attributes composing the optimal isomorphism between the *FindRequestedItem* and the *LocateVehicle* transactions.

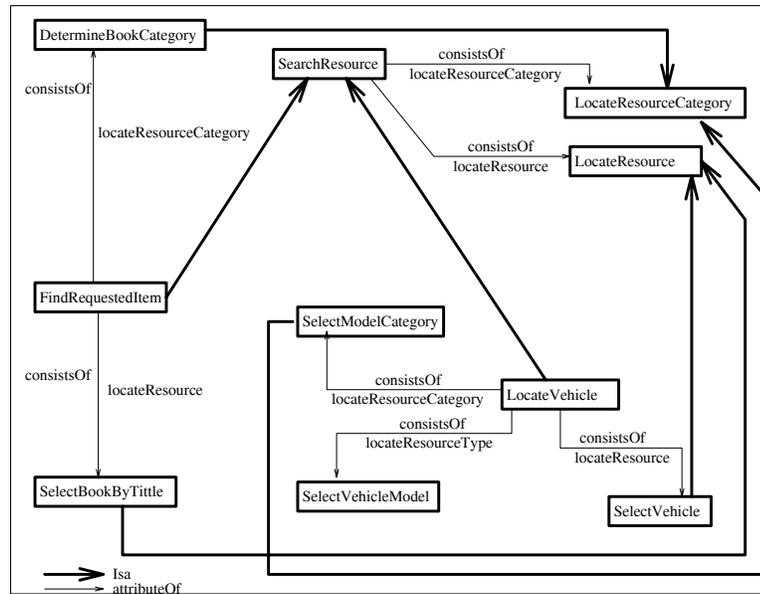


Figure 6.11: Generalization of the *FindRequestedItem* and *LocateVehicle*

The presence of all these generalization relations would affect the analysis of the similarity between the *FindRequestedItem* and the *LocateVehicle* transactions in two ways.

First, they would allow similarity analysis to converge sooner to the final optimal isomorphism. This would result from the immediate mapping between the attributes *locateResourceCategory* and the attributes *locateResource* of the transactions under comparison. Both these mappings would be undoubtedly elements of the optimal isomorphism between the transactions *FindRequestedItem* and *LocateVehicle*, since they would map attributes sharing the same original class (i.e. the attribute classes *locateResourceCategory* and *locateResource* of the transaction *SearchResource*) according to theorem 3.10 in chapter 3. Hence, instead of having to evaluate 6 different isomorphisms between the decomposing attributes of the involved transactions, it would only be necessary to evaluate the distance of the single possible isomorphism, deduced from that theorem.

Also, the presence of the common generalizing transaction *SearchResource* would increase the overall similarity estimate between the involved transactions. For instance, the absolute - and therefore the relative - generalization distance, estimated as the sum of

the inverse depths of the non common superclasses of the *FindRequestedItem* and *LocateVehicle* transactions, would decrease to 0.66 (i.e. $1/3 + 1/3$) from its original value of 1 (i.e. $1/2 + 1/2$).

These observations indicate that the detection of analogies is tolerant to specific forms and degrees of utilization of the generalization and the classification abstractions. This behavior becomes the main empowering aspect of our approach and justifies our claim that as a computational model of analogy, analogical similarity is independent from domain specific types of knowledge and that it is not as knowledge acquisition intensive as other models.

6.3.2 Example 2: Similarity Based Modeling of Specifications

Our next example illustrates how similarity and prototypicality may be utilized in modeling specifications by reuse.

Assume that we need to specify a software system for a university library. This system could be specified as an aggregation of transactions representing its main functions, entities representing the main kinds of information kept and/or required by the system for carrying out its functions and agents who interact with the system according to the *system* and the *usage world* perspectives, which discussed in [MBJK90].

Here, we focus on how to specify the transactions of the university library system by reusing specifications for transactions of other systems, supporting the hiring of resources, such as a *Municipal Library System*, a *Car Rental System* and a *Motorbike Rental System*. These specifications are assumed as being stored in some repository and described according to an extended version of the transactions specification model that was defined in our first example(see figure 6.12).

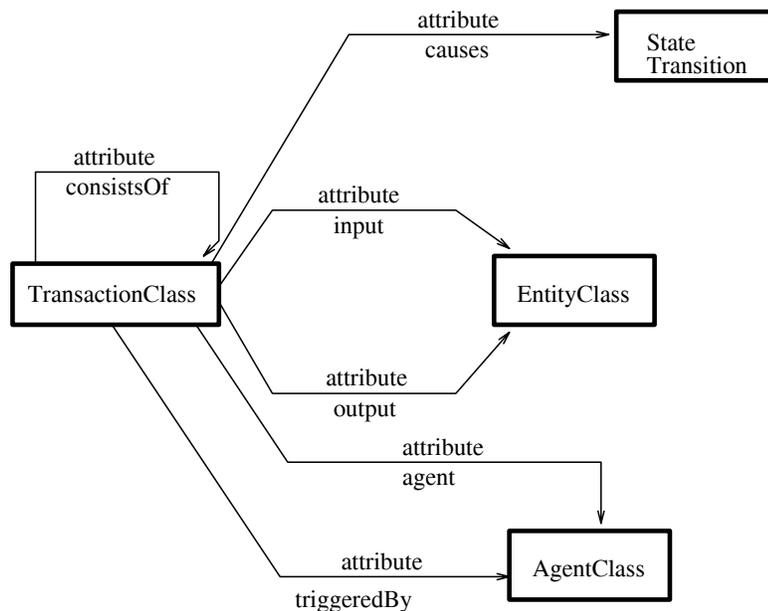


Figure 6.12: Extended Model for Describing Transactions

As it can be observed from this figure, the kernel meta model of describing transactions in terms of its inputs, outputs and decomposing transactions, that was presented in figure 6.5, is augmented by attributes depicting the agent who is responsible for executing it (i.e. the *triggeredBy* attribute), the agent on behalf of whom the transaction is carried out (i.e. the *agent* attribute) and the possible state transitions that may be caused to resources managed by the relevant information system, as a result of the transaction's execution (i.e. the *causes* attribute).

Let us assume that our purpose is to specify some transaction for returning books from borrowers back to the library (referred to as the *ReturnLibraryItem* transaction in the following).

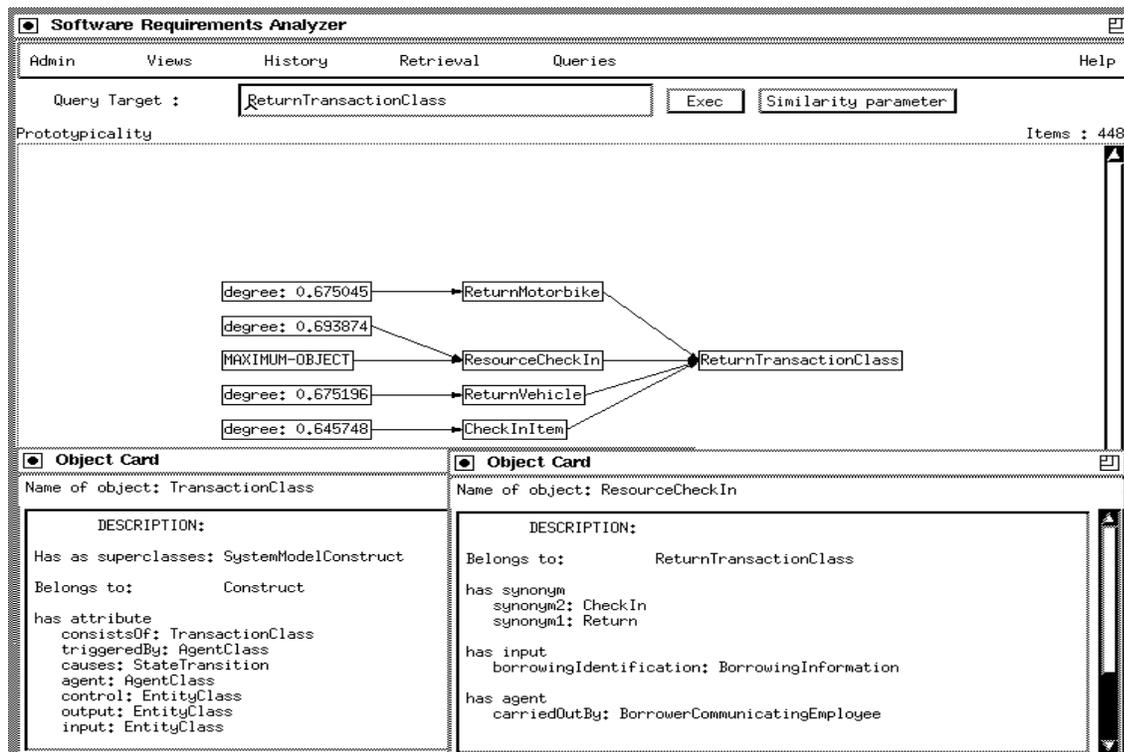


Figure 6.13: Selection of Prototypical Examples

Initially, the software engineer has to provide a rough specification for this transaction, according to the transaction specification model. As the basis for prototypicality estimates, similarity analysis may help the engineer by assigning prototypical examples (i.e. instances) of the model for specifying transactions. This aid may prove particularly useful in cases where the software engineer is not familiar with the abstract specification

model and needs some guidance for using it. As pointed out by relevant cognitive studies[MS92], the presentation of concrete model examples contributes significantly to understanding the respective models.

Prototypicality is estimated as the average similarity of some specification with the other specifications built according to a given model. Figure 6.13 presents the prototypicalities of the transactions supporting the return of borrowed resources, from their borrowers back to the resource holding system. The most prototypical transaction *ResourceCheckIn* can be used as an example of how to specify such transactions.

Once at least a partial specification of the *ReturnLibraryItem* transaction exists (i.e. when the dashed part of the innermost spiral in figure 6.1 has been completed), the software engineer can look for other transactions similar to it. Given a set of such transactions ranked in descending similarity order (see figure 6.14), the engineer can focus on successively less similar specifications and try to reuse elements from them.

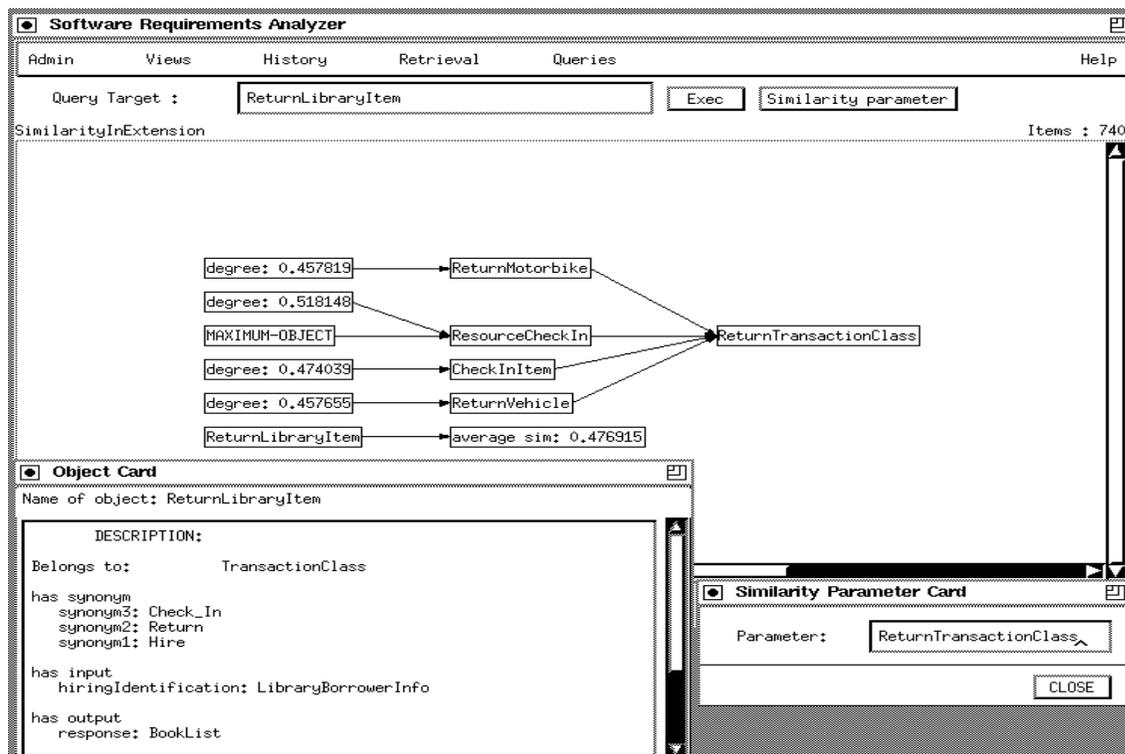


Figure 6.14: Ranked Retrieval By Similarity

The detailed similarity analysis between any such specification and the one under construction, results in a distinction between their analogous and unique elements(see the

similarity analysis graph of figure 6.15). As already discussed, this distinction may support the completion of a target specification in three distinct ways. The unique elements of the source specification may indicate an incompleteness in the target, and therefore they should be examined for reuse. For instance, the agent who is responsible for the execution of the *ReturnLibraryItem* transaction is not specified, unlike the agent who is responsible for the execution of the *CheckInItem* transaction (see *carriedOutby* attribute of the transaction *CheckInItem* in figure 6.15).

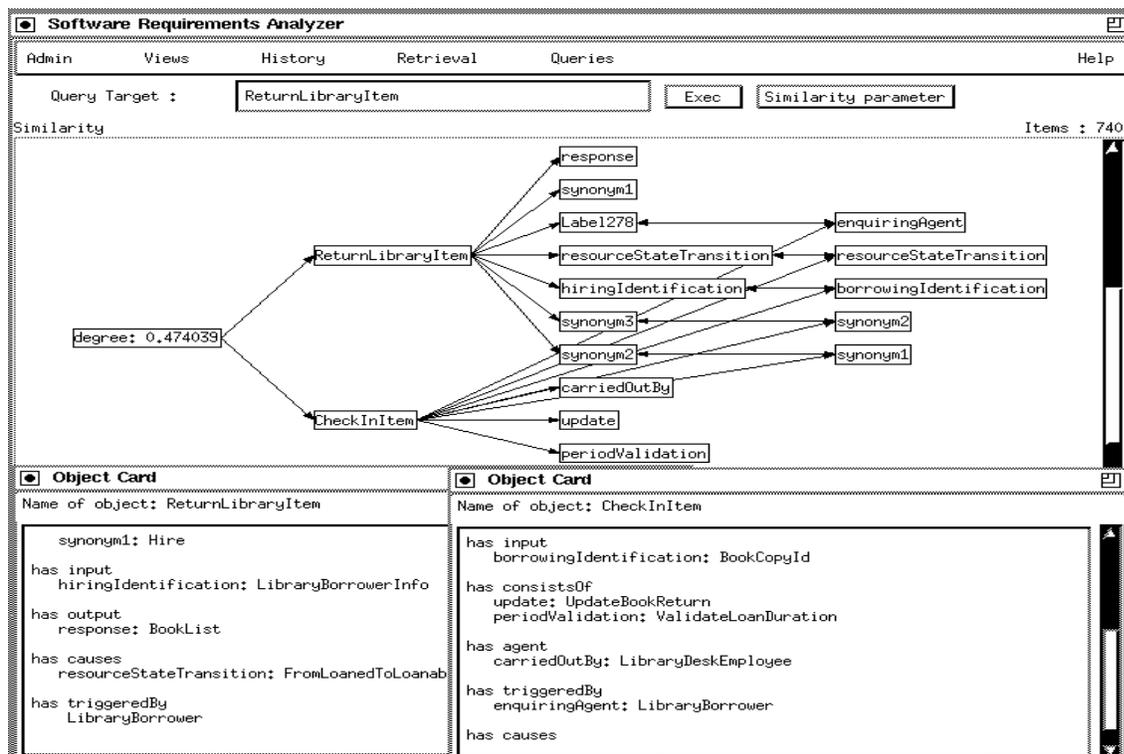


Figure 6.15: Similarity Between Transactions

On the other hand, both these transactions have attributes identifying a past resource borrowing activity (i.e. attributes *borrowingIdentification* and *hiringIdentification*), albeit in different ways. The *ReturnLibraryItem* exploits some general borrower information, unlike the *CheckInItem* transaction, which uses a unique code assigned to each copy of a book. Such analogous but slightly different elements may give rise to alternative solutions. The use of information about a borrower, enables the immediate retrieval of all the books he has borrowed. The use of book-codes identifies only one of these books. If all the checked out books were required there should be predicted an appropriate decomposition of the transaction so as to retrieve possible other borrowed books, in a second

step of operation.

Notice also that elements which are unique to the target specification could be redundant and thus their presence should be further justified (e.g. the attribute *response* of *ReturnLibraryItem*).

Although rough as a scenario, these cases illustrate how similarity analysis can support the modeling of new specifications by reusing elements from existing specifications. In fact, they provide just an initial understanding of the process of analogical reuse. We strongly suggest that in spite of the aid in detecting analogies, human cooperation is inevitable in exploiting detected analogies and differences in completing specifications. Redundant and missing elements must be verified by the engineers, who are developing the relevant specifications. Also, alternative modeling of analogous elements can only be decided and carried out by them.

6.3.3 Example 3: Applicability of Similarity to Further Specification Models

Our next example, taken from [Maid92], shows that similarity analysis can be applied to requirements specifications described according to yet another meta model. This applicability results from the capability of the semantic modeling abstractions assumed in our framework, to define different specification models as meta models and then to represent particular specifications as instances of these meta models.

The selected example exposes an analogy between two different application domains. The first of them is the domain of *air traffic control*(ATC) and the second is the domain of *flexible manufacturing systems*(FMS). The essence of the analogy between these two domains has been extensively discussed in [MS91,Maid92]. In summary, both these domains involve objects (i.e. aircrafts and products) moving in specific spaces(i.e. air corridors and conveyor belts) according to movement plants(i.e. flight plans and production plants)[Maid92].

The ATC and the FMS domains can be specified according to a meta model of expressing knowledge about software engineering domains, defined in [Maid92]. A subset of this meta model, sufficient for representing the selected domains, consists of the following typed predicates:

```

objectstructure(Object,Object,structural_relation)
statetransition(object,source,destination,StateTransition)
objecttype(Object,ObjectType)
function(function,object,source,destination,StateTransition)

```

These predicates are interpreted[Maid92] as follows:

objectstructure predicate: It defines the membership of its second argument, in the set expressed by its first argument. The third argument expresses a cardinality constraint on the set membership relation.

statetransition predicate: It defines a state transition, named by its fourth argument, concerning the object represented by its first argument. This transition is perceived as a change in the set membership of the concerned object from the set expressed by the second argument of the predicate to the set expressed by the third argument.

objecttype predicate: It defines the typing of the first argument as an instance of the type expressed by its second argument.

function predicate: It defines causal relations between functions, which are described by its first argument and the state transitions they cause. These state transitions are expressed by the second, the third, the fourth and the fifth arguments of the predicate exactly like the *statetransition* predicate.

Figure 6.16 presents the formal definition of these predicates in the notation introduced in the third chapter. The original names, employed by the author of the model, are used instead of internal identifiers for the sake of readability.

A few remarks are necessary for understanding the rationale behind this transformation, which is also graphically presented in *figure 6.17*

First, predicates that express directed relations are defined as attribute objects. These attributes are associated with the objects, which are the origins of a relation, and have as values the objects, which are its destinations. The only exception is the definition of *Objects* as instances of *ObjectTypes*, since the classification relation in our framework corresponds to the typing relation conveyed by the predicate *objecttype*. Also, predicate arguments characterizing relations as a whole, rather than any of their arguments (such the *structural_relation* argument of the predicate *objectstructure*, which expresses the

$$\begin{aligned}
& [ObjectType, ObjectType, nil, \{ \#C3, \#C, \#I \}, \{ \#C2 \}, \emptyset, nil] \\
& [Object, Object, nil, \{ \#C2, \#C, \#I, ObjectType \}, \{ \#C1 \}, \{ function, objectstructure \}, nil] \\
& [function, function, Object, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \emptyset, Function] \\
& [objectstructure, objectstructure, Object, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \{ structural_relation \}, Object] \\
& [structural_relation, structural_relation, objectstructure, \{ \#C1, \#C, \#A \}, \{ \#C0 \}, \emptyset, Structural_Relation] \\
& [Structural_Relation, Structural_Relation, nil, \{ \#C2, \#C, \#I \}, \{ \#C1 \}, \emptyset, nil] \\
& [Function, Function, nil, \{ \#C2, \#C, \#I \}, \{ \#C1 \}, \{ causes \}, nil] \\
& [causes, causes, Function, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \emptyset, StateTransition] \\
& [StateTransition, StateTransition, nil, \{ \#C2, \#C, \#I \}, \{ \#C1 \}, \{ object, source, destination \}, nil] \\
& [object, object, StateTransition, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \emptyset, Object] \\
& [source, source, StateTransition, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \emptyset, Object] \\
& [destination, destination, StateTransition, \{ \#C2, \#C, \#A \}, \{ \#C1 \}, \emptyset, Object]
\end{aligned}$$

Figure 6.16: Formal Definition of a Meta Model for Describing Domain Abstractions

cardinality of the containment of objects in object structures), were defined as attributes of the attributes representing the relevant relations.

The definition of the original meta model in terms of the three semantic modeling abstractions of our framework, allows the specification of the ATC and the FMS domains as instances of this meta model in the same framework (see *figure 6.18*).

Similarity Analysis of the ATC and FMS Domains. The analysis of similarity between the ATC and the FMS domains is possible after their description in terms of the semantic modeling abstractions, in our representation framework. Very important to this analysis is the classification of the relations between the concepts of these domains under different attribute classes of the defined meta model.

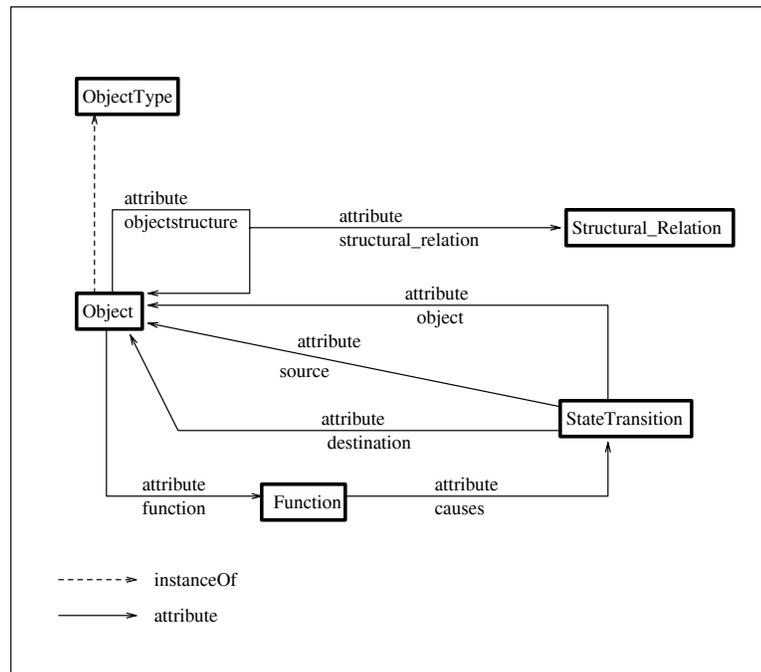


Figure 6.17: Graphical Representation of the Meta Model for Describing Domain Abstractions

In particular, the classification of the attributes of instances of the class *Object* as instances of the attribute classes *objectstructure* and *function*, leads to a single possible isomorphism, due to the semantic homogeneity criterion.

This isomorphism, which maps the attributes *hasSpace* and *monitors* of the ATC and the FMS domains, onto each other, dictates the analogy between the air spaces in the ATC domain and the track sections in the FMS domain, as well as the analogy between their monitoring functions. Similarly, the isomorphism, which maps the attributes *affectedObj* of *AircraftTransition* and *ProductTransition*, extends the analogy, by expressing the analogous roles of the aircrafts and the products. It is clear that the complete analogy between the involved domains, is revealed only by considering all the successive isomorphisms between the attributes in the closures of their attribution graphs.

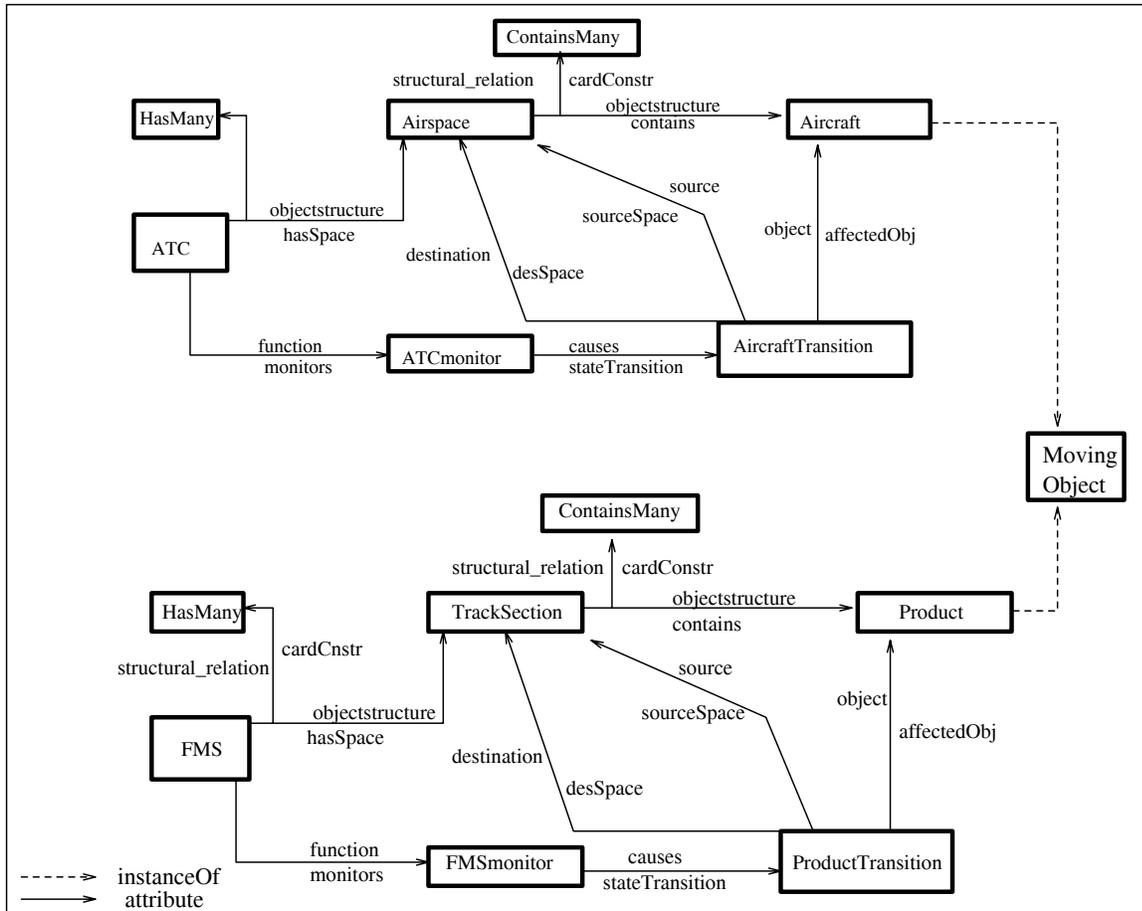


Figure 6.18: Specifications of the ATC and FMS domains

Similarity analysis detected this analogy without it being necessary to match the specifications of the domains to any common abstraction, as in [Maid92], because of their particular structuring according to the prescribed meta-model, which enabled the semantic homogeneity criterion to dictate single possible isomorphisms. As in our previous example, the incorporation of an *object monitoring abstraction* as in [Maid92], abstracting the common aspects of the ATC and the FMS domains (see figure 6.19) would increase their overall similarity measure, and thus the aptness of the detected analogies. However, it would not alter the evaluated isomorphisms between the involved domains.

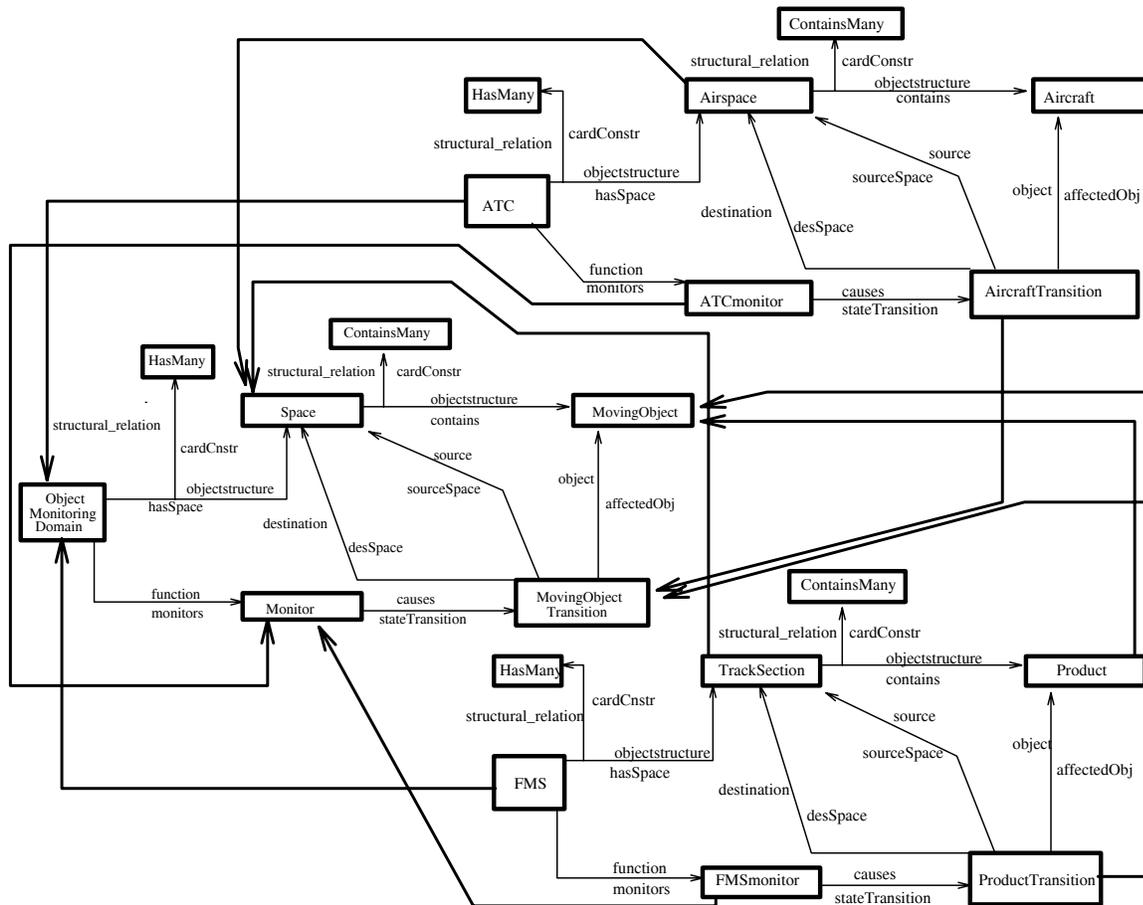


Figure 6.19: The Object Monitoring Abstraction of the ATC and FMS Domains

6.4 Similarity and Other Models for Analogical Reuse of Specifications

In the previous sections, we demonstrated through examples, how similarity analysis computationally supports the analogical reuse of specifications. In this section, we discuss the merits of this and other computational models of analogical reuse, with respect to certain pragmatic restrictions, for reusing requirements.

6.4.1 The Pragmatic Restrictions of Analogical Reuse of Specifications

The real challenge for computational models for analogical reuse of specifications is:

- to support reuse both across different application domains and within the same domain (i.e. *horizontal* and *vertical* reuse in the terminology of Diaz[Diaz93]);

- to be as tolerant as possible to the knowledge acquisition bottleneck; and,
- to be applicable in a wide spectrum of different representations of reusable specifications.

This wish-list summarizes certain predictions and observations about the pragmatics of requirements reuse.

First of all, reuse across domains, is expected to result into higher payoffs than reuse within domains[Maid92,Diaz93]. The diversity and the size of the population of artifacts that can be accumulated from systems in different domains, make more likely the location of artifacts useful in future development sessions. However, reuse across domains involves harder operational problems(i.e. retrieval, comprehension, modification) primarily due to the lack of adequate interdomain abstractions to support the whole process[Kru92].

This requirement has certain implications for the criteria of mapping, that can be employed by computational models of analogical reasoning. Systems forming mappings on the basis of knowledge preserving criteria are inherently domain dependent, since such knowledge(referred to as *determinants* in [Rus88]) is rarely meaningful across domains. On the other hand, systems which are based on syntactic aspects of representations of specifications can be domain independent.

Another pragmatic objective is to avoid to augment the knowledge acquisition bottleneck of the requirements engineering process[Hof93]. This bottleneck would be more severe if software engineers had to incorporate in their specifications additional relationships, supporting solely the detection of analogies but having no other apparent effect to the main role of these specifications, which is the accurate description of the client's objectives.

Finally, the highest possible independence from particular meta models for expressing specifications is motivated by a purely pragmatic concern. It is evident from recent surveys of methods and tools for requirements[Hof93,Pohl93] and computer aided software engineering[Fug93], that there exists a plethora of models and languages for specifying requirements(e.g. dataflow and entity-relationship diagrams[AG90,Mal92], structure diagrams[Cam86], statecharts[Fug93], activity and information precedence graphs[LGN81], Z[Spiv90]). This diversity of requirements specification models and languages has been plausibly attributed[JFH92] to the multitude of types of requirements

knowledge and the need to allow software engineers elicit and express requirements in a manner familiar to them. Models for analogical reuse, which are sensitive to particular specification models or languages, will inevitably have a limited pragmatic utility.

In the following, we survey the current approaches to analogical reuse of specifications and argue about their limited ability to satisfy the previous requirements.

6.4.2 Existing Approaches to Analogical Reuse

Analogical reasoning as a paradigm for software reuse, has not yet received a large attention in the literature. According to our knowledge, the approaches which computationally realize this paradigm include the IRA system[MS91,Maid92,MS92], and the SPECIFIER system[MH91a,MH91b]. Also, there have been proposed systems and methods, which incorporate forms of case-based reasoning in supporting reuse, such as KAPTUR and LEARN[BM91], the Techne project's approach[MR91] and the genericity approach[Kat94].

The Intelligent Requirements Advisor. The Intelligent Requirements Advisor(IRA system) is a special purpose analogical reasoner developed to support analogical reuse of requirements specifications. The system is based on a predefined set of abstractions aggregating aspects of specifications, which determine the presence of useful analogies.

Abstractions aggregate sets of objects, state transitions of, categorizations and other relations between them. An example of such abstractions is the *Basic Renewable Resource Management*, which models stores of resources that are distributed to various sinks[Maid92]. Resources are periodically replenished, when their quantity in the store reaches a minimum level. Instantiations of this abstraction include bank accounts and department stores.

Elaboration and retrieval in IRA depends on these abstractions. For instance, the analogy between bank accounts and department stores, would be revealed by matching each of them to their common abstraction. One of these instances will be retrieved as a source analog if the other instance, serving as a target, matches successfully the relevant abstraction. Matching is based on the syntactic criteria of *consistency* and *systematicity* of the Structure Mapping theory[Gen83](see also chapters 2 and 7 in this thesis).

The ability of IRA to detect analogies is inevitably bounded by its determinant abstractions. Certainly, the completeness of these abstractions is still an open research issue, as even the designers of the system admit[Maid92].

Furthermore, the practical utility of the system is limited due to the adoption of a particular meta-model of relationships and entity structures, for specifying requirements(see section 6.3.3). These relationships and entity structures do not correspond to either general semantic modeling abstractions, or to widely accepted constructs for modeling requirements, and thus it is very unlikely for them to gain a high practical utilization.

Nevertheless, it must be pointed out that IRA has made a significant contribution to the problem of devising well articulated abstractions for upstream software artifacts such as requirements specifications.

The SPECIFIER system. SPECIFIER[MH91a,MH91b] supports the completion and formalization of natural language specifications of abstract data types and programs by abstraction or analogy-driven derivations. Abstraction-based derivations are based on *operation schemas*, which formally specify operations over data types. Operation schemas are indexed by representative concepts, which are amenable to the relevant operations. SPECIFIER attempts to interpret the concepts present in a user's specification of some program or data type, in reference to some of these representative concepts. Subject to a successful interpretation, it retrieves the associated operation schemas and instantiates them, according to their own instantiation rules.

If this mode of derivation fails, SPECIFIER attempts to establish an analogy between an internal structural description of a user's specification and stored structural descriptions of representative concepts. It does so using predefined primitive relations determining pairs of analogous concepts. Analogies are utilized for importing and instantiating schemas associated with the source representative concepts. This instantiation constitutes the formalization of the user's specification.

The reliance of SPECIFIER on a-priori asserted relations of primitive analogies between concepts is its main drawback. The acquisition of such relations may be feasible for abstract data types and operations but it becomes an extremely labour-intensive task, considering the diversity of concepts that may exist in specifications of fairly complex systems.

The Techne project. The Techne project[MR91] concerns the broader problem of providing knowledge representation and management solutions adequate for reusing knowledge of existing information systems in developing new ones. In this framework, reuse is realized as a form of case-based reasoning, focusing on the problem of adapting previous cases of a software design problem to a current situation.

The KAPTUR system. The KAPTUR system[BM91] adopts the indexing approach to case-based reasoning. Thus, it selects reusable artifacts based on their organization around discriminating features. This organization is based on pattern matching techniques over a special vocabulary. The LEARN system[BM91] acts as an apprentice, operating under the supervision of an expert. It attempts to generalize solutions to software design problems(e.g. a design repairing rule) from single examples, using case-based reasoning and explanation-based learning. Subject to successful generalization, LEARN is also capable of applying a general solution to a different input problem.

The Genericity Approach. The genericity approach[Kat94] is based on a hierarchy of parametrized abstract data types and operations, called *the genericity hierarchy*, which supports the retrieval and the adaptation of source code artifacts, based on high level specifications.

Selection is performed through four different matching modes (i.e. exact, generic, similarity and optional match), which are applied successively, subject to previous failures to locate a component. The non exact modes of matching(i.e. generic, similarity and optional match) rely on a-priori asserted relations between concepts that constitute specifications(e.g. the *substitution relation*[Kat94] for generic matching, the *modification relation*[Kat94] for similarity matching).

This approach may be bounded by the cost of acquiring such relations, for concepts in specifications of complex software systems. However, its feasibility has been demonstrated for artifacts such as abstract data types. Also, it addresses the modification problem of reuse, through a cooperation with the user.

As evidenced by this short survey, the current approaches to analogical reuse of specifications, fail to overcome the pragmatic restrictions that may limit the utility of the paradigm. This justifies the application of analogical similarity as an alternative, which can overcome these problems to a certain extent.

6.4.3 The Merits of Similarity

The utilization of similarity analysis as a model supporting the analogical reuse of requirements specifications, was demonstrated through specific examples in section 6.3.

Similarity analysis is able to meet, to a certain extent, all of the pragmatic concerns for such models.

It is able to support intra and inter domain analogical reuse. This ability results from the criteria utilized for detecting analogies between objects (i.e. the semantic homogeneity and the minimum distance isomorphism), which are both defined in terms of three domain independent semantic modeling abstractions (i.e. classification, generalization and attribution).

Similarity analysis is also applicable to specifications devised according to a wide class of specification models. This class includes all those specification models, which can be defined using the prescribed semantic modeling abstractions. Of course, the real pragmatic utility of the model, with respect to this aspect depends on the availability of tools, transforming automatically specifications expressed in the original syntax of such models into equivalent specifications, in the representational framework of the similarity model.

Furthermore, since the model estimates distance and salience measures requiring no special knowledge acquired for these purposes it is completely domain independent and, realizes the problem of the knowledge acquisition bottleneck in requirements engineering.

6.5 Similarity Analysis for Viewpoint Integration

6.5.1 The Problem of Viewpoints Integration

Another application of similarity analysis regards the problem of viewpoint integration. Viewpoints are specifications of requirements, which are elicited and modeled from different perspectives[LF91,NKF93], held by the different agents, who participate in the process of engineering requirements, especially for complex and composite systems. These agents include the clients in the application domain, who express their objectives about the system to be built and the software engineers, who produce specifications as conceptual models of these objectives. Overlaps, complementary aspects and contradictions between different viewpoints must be identified and resolved(i.e. *viewpoints integration*). Viewpoints integration is necessary or otherwise, requirements engineering will not result into a complete, consistent and agreed upon specification, expressing as accurately as possible the objectives of the system's clients.

Integration involves, two successive stages. The first of them concerns with how to detect discrepancies and resemblances between different viewpoints(i.e. *analysis of viewpoints*). The second deals with how to merge both the resembling and the distinct elements, identified in the previous stage(i.e. *viewpoint merging*). Viewpoints analysis is amenable to computational reasoning mechanisms, capable of detecting discrepancies and similarities(see [LF91]). Unlike it, viewpoint merging can hardly be automated since it is not merely a matter of knowing the differences and the resemblances between the elements of different viewpoints. In fact, it has to take into account further criteria, such as the comprehensibility of the final specification, its adequacy for the consequent stages of the system design and implementation or even the maintainability of the system in the long term.

In the following, we discuss how similarity analysis can be used as a computational reasoning mechanism for viewpoint analysis. We, like other authors[LF91], assume that the detection of analogies between elements of viewpoints offers a reasonable basis for negotiating their merging. Before elaborating into the details of the application, we distinguish between the general possible solutions to viewpoint analysis.

6.5.2 General Approaches to Viewpoint Analysis

The basic problem in viewpoints analysis is to decide whether different elements in different viewpoints express the same underlying entity (e.g. an objective about the system, an aspect of its usage environment etc.). Two kinds of knowledge may be used in making such decisions, these are *grounding* and *meta* knowledge.

Grounding knowledge relates directly elements in viewpoints with terms in a vocabulary expressing concepts and relations in some domain. Subject to the ability to express M:1 such relations between viewpoint elements and vocabulary terms, grounding knowledge can lead to a definite solution to the viewpoint analysis stage. However, this approach does not scale up well with complex domains. This is because humans, even if they are experts in some domain, use large and diverse vocabularies of items to denote the vast population of concepts and interrelationships [FLGD87]. Moreover, different vocabularies must be acquired and maintained for different domains.

The *meta* knowledge approach suggests the classification of viewpoint elements under well-defined, general modeling concepts, independent from particular application domains, which express general semantic properties. This classification may then be used for comparing viewpoint elements. Elements sharing the same semantic properties are likely to refer to the same underlying entity. Such a comparison can be performed according to *logical equivalence relations* as in [GB91, TS93] or be a form of inexact matching [FKN91]. Notice that, subject to the genericity of the classificatory concepts, elements with identical semantics may refer to different real-world entities. Also, comparisons based on logical equivalence relations may not lead to a unique solution, if they result into M:N associations between elements of distinct viewpoints.

The meta knowledge approach is more feasible than the grounding knowledge approach from a pragmatic point of view. This is because the number of the classificatory concepts is significantly smaller than the number of the grounding terms and moreover, subject to their well-defined semantics, these concepts will not be as ambiguous as the single grounding terms.

As a computational solution to the viewpoint analysis problem, the application of similarity analysis can be classified under the meta knowledge approach. This is because similarity analysis, in this particular application, is informed by the classification of

viewpoint elements under a particular meta model abstracting general properties of concepts and relationships, which have been recently discussed in the conceptual modeling literature[MP93,Sto93]. Viewpoints are realized as conceptual models, which are expressed either directly in terms of the semantic modeling abstractions constituting the representation terrain of similarity analysis or according to specification models, whose constructs can be defined through these abstractions. Furthermore, they are classified under the appropriate classes of the viewpoint analysis meta model.

The following three sections, introduce this meta model, discuss why the similarity model is inherently capable for performing viewpoint analysis and give examples of detecting analogies and discrepancies between viewpoints.

6.5.3 The Meta Model

The meta model is organized as an Isa taxonomy of classes grouping components of conceptual models according to domain and specification model independent properties of concepts. Such properties have been discussed in the semantic modeling literature[Sto93,MP93]. Some of them have also been identified in elementary set-theory.

The meta model is by no means complete. It is only a first attempt to demonstrate how similarity analysis can be used for comparing viewpoints, exploiting domain and specification model independent meta knowledge about them.

The most general class of the meta model is the class *ConceptModelingComponent*. This class groups all the components of conceptual models, regardless of the real world concept that is modeled by them. It also abstracts two basic aspects of modeling constructs of data models(i.e. a particular type of specification models), which, as shown in figure 6.20, include: (a) the attachment of attributes(i.e. the attribute *aggregates*) and, (b) the presence of identifiers(i.e. the attribute *identifiedBy*).

The *ConceptModelingComponent* class is partitioned in two specializations, the *EntityModelingComponent* and the *RelationModelingComponent*, which include the elements in conceptual models that refer to entities and relations, respectively.

General Classes of Components Modeling Entities. The *EntityModelingComponent* class is further partitioned into six subclasses, shown in figure 6.21.

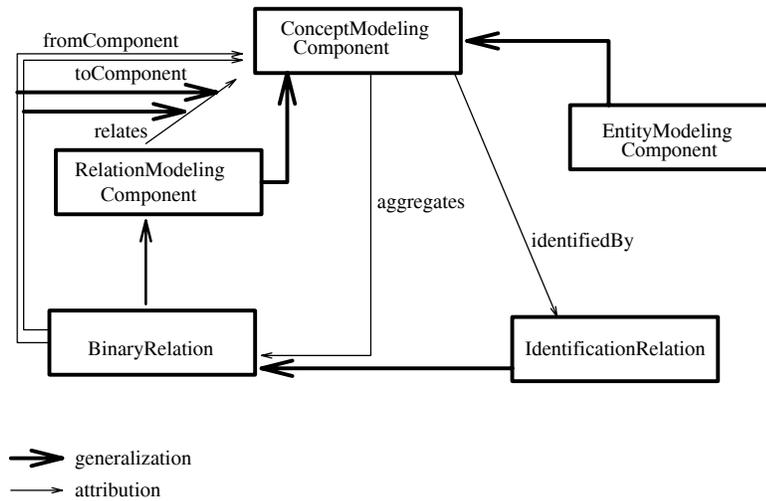


Figure 6.20: Components Expressing Entities and Relations

The class *NaturalKindComponent* groups components representing physical entities, as opposed to tailor made or invented entities [Smi89], which are represented by components grouped under the class *NominalKindComponent*.

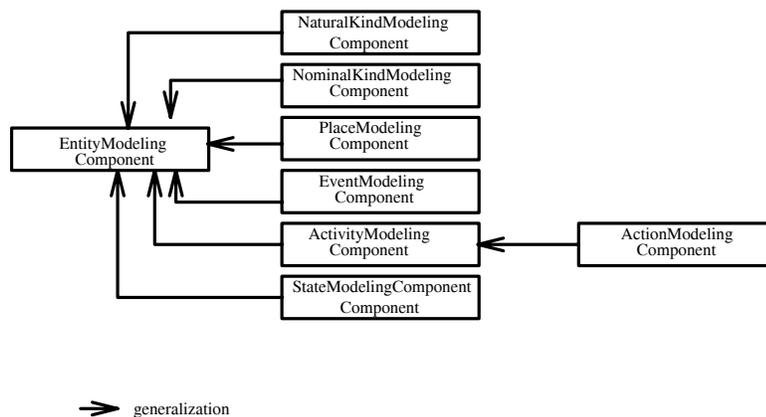


Figure 6.21: Partition of Entity Modeling Components

The class *PlaceComponent* groups components modeling locations. The class *EventComponent* groups components modeling events(i.e. everything in the real world occurring at a specific place and time or time period). The class *StateComponent* groups components reflecting states. States are bundles of co-occurring properties of real world concepts, which together denote a special situation for them. The class *ActivityComponent* groups components representing processes in the real world. This class is specialized by the class *ActionComponent*, which groups components modeling actions taken by some natural or nominal kind agent, with a specific purpose.

General Classes of Components Modeling Relations. Like the *EntityModelingComponent*, the class *RelationModelingComponent*, which groups components expressing relations of any arity, is further specialized. First, components of conceptual models that express directed binary relations are grouped under the class *BinaryRelation*, which has two attributes representing the arguments of such relations(i.e. *fromComponent*, *toComponent* in figure 6.20).

Binary relations are further grouped into classes reflecting cardinality constraints, general mathematical properties, existential constraints and other modeling semantics(see figure 6.22).

Cardinality constraints, are reflected by the following seven classes:

- (1) the class *OneToOneRelation*, which groups components modeling 1:1 relations,
- (2) the class *ManyToOneRelation*, which groups components modeling N:1 relations,
- (3) the class *ManyToManyRelation*, which groups components modeling N:M relations,
- (4) the class *OneToManyRelation* (grouping components modeling 1:N relations),
- (5) the class *TotalRelation*, which groups components modeling relations, in which every element in their domain must be associated with some element in their range,
- (6) the class *OptionalRelation*, which groups components modeling relations, in which it is not necessary for every element in their domain to be associated with some element in their range and,
- (7) the class *OntoRelation*, which groups components modeling relations, in which every element in their range must be associated with an element in their domain

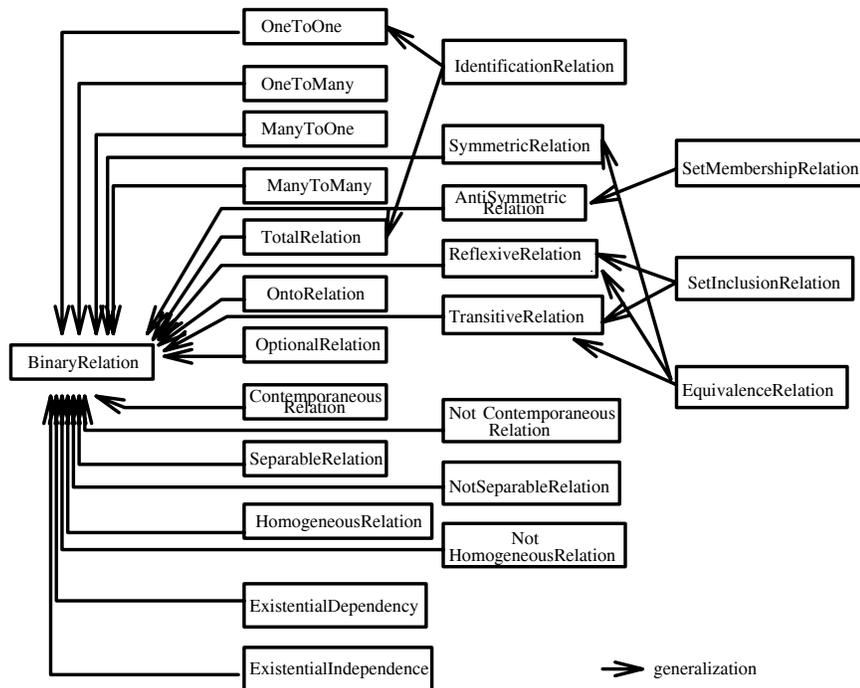


Figure 6.22: Special Kinds of Binary Relations

A second group of subclasses reflects mathematical properties of relations. This group includes the classes:

- (1) *SymmetricRelation*;
- (2) *AntiSymmetricRelation*;
- (3) *ReflexiveRelation*; and,
- (4) *TransitiveRelation*.

These four classes group components modeling relations whose domains and ranges are identical and moreover are symmetric, antisymmetric, reflexive and transitive, respectively.

A third group of subclasses, capture existence constraints that may be implied by binary relations. This group includes the classes:

(1) *ExistentialDependencyRelation*, which groups components modeling binary relations where the existence of the *toComponent* value depends on the existence of the *fromComponent* value and,

(2) *ExistentialIndependencyRelation*, which groups components modeling binary relations where the existence of the involved elements does not depend on to each other

Six more specializations of the *BinaryRelation* class enable - in pairs - three orthogonal partitions, according to criteria discussed in [Sto93]. The first of these criteria concerns whether the arguments of some relation must temporarily coexist or not. Components modeling relations whose elements must coexist are grouped under the class *ContemporaneousRelation* (e.g. *committee hasMember consultant*), while components modeling relations whose elements must not coexist, are grouped under the class *NotContemporaneousRelation* (e.g. *ancientArtifact hasBeenStudied By archaeologist*, since the entire ancient artifact may not exist at the time the archaeologist studies it based on some of its parts).

The second criterion concerns, whether the arguments of a relation are at least of one common kind or not. If they are, the components modeling the relevant relations are grouped under the class *HomogeneousRelation*, while if they are not, the relevant components are grouped under the class *NotHomogeneousRelation*. It must be pointed out that the "one common kind" condition makes sure that the involved concepts will share a bundle of common properties implied by this kind (e.g. *bread has slice*).

The third criterion distinguishes between relations, whose arguments can be physically disconnected and relations, whose arguments cannot. The former relations are grouped by the class *SeparableRelation* (e.g. *bicycle hasPart wheel*) and the latter by the class *NotSeparableRelation* (e.g. *bicycle isMadeOf aluminum*).

Finally, there are a few more domain and specification model independent special kinds of binary relations, grouped by the classes:

(1) *EquivalenceRelation* (i.e. the class of reflexive, symmetric and transitive relations);

(2) *SetMembershipRelation* (i.e. the class of relations with a set membership semantics);

(3) *SetInclusionRelation* (i.e. the class of relations with a set inclusion semantics); and,

(4) *IdentificationRelation* (i.e. the class of 1:1 and total relations enabling the unique identification of their *fromComponent* by their *toComponent*).

Specification Model Dependent Extensions of the Meta Model. This meta model of domain and specification model independent entity and relation types is further specialized by specific constructs of different viewpoint specification models.

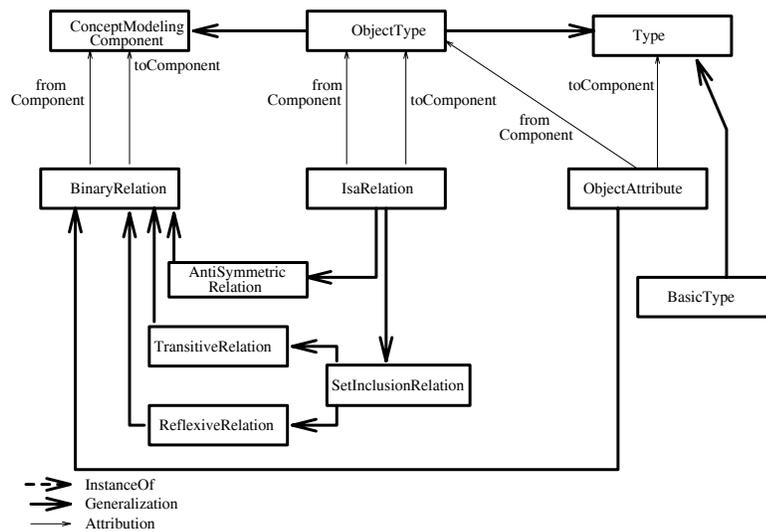


Figure 6.23: *Isa Relations in Object-Oriented Specification Models*

Figure 6.23 presents how the isa relation supported by a hypothetical object-oriented data model can be expressed as a subclass of particular classes of binary relations (i.e. the class *IsaRelation*). *IsaRelation* is declared as a subclass of the *SetInclusionRelation* to express that isa relations in the particular data model have a set inclusion semantics. It is also declared as a subclass of the *AntiSymmetricRelation* class to express that an object type in the hypothetical object-oriented data model can not be a specialization of itself. Furthermore, the *IsaRelation* restricts the attributes *fromComponent* and *toComponent* inherited from the class *BinaryRelation* to take as values only object types of the hypothetical data model. This declaration precludes isa relations between components in conceptual schemas, which represent attributes.

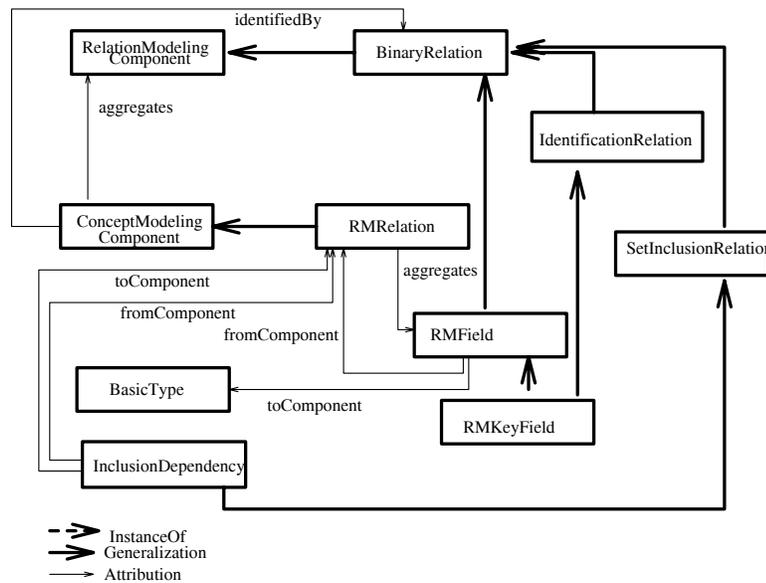


Figure 6.24: Relations in Relational Specification Models

Figure 6.24 presents how constructs of the relational data model [Codd70,Codd79], such as *relations*, *fields* and *inclusion dependencies* are defined as subclasses of general classes of the viewpoint analysis meta model. The class *RMRelation* has been defined as a subclass of the class *ConceptModelingComponent*, indicating that relations in relational conceptual schemas can express both relationships and entities. The class *RMRelation* restricts the attribute *aggregates* inherited from the class *ConceptModelingComponent* to take values in the class *RMField*, which expresses the fields in the relational data model. Since fields express binary relations, the class *RMField* is declared as a subclass of the class *BinaryRelation*. Finally, inclusion dependencies are represented by the class *InclusionDependency*. Due to the set inclusion semantics of such dependencies in the relational model, this class is declared as a subclass of the *SetInclusionDependency* class.

These examples of representing constructs of particular specification models as subclasses of the general classes of the meta model, illustrate how the elements of viewpoints expressed according to such specification models can be automatically classified under these general classes.

6.6 Similarity Analysis as a Means for Viewpoint Analysis

Similarity analysis can be used for comparing viewpoints, built according to the same or different specification models, provided that their elements are classified under the classes of the introduced meta model. Notice that part of this classification may be done automatically in cases of elements, which are modeled according to specification model constructs, that specialize classes of the meta model. However, the rest of the classification has to be carried out manually.

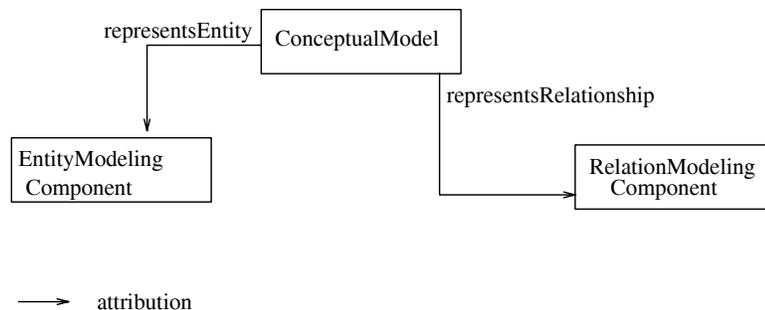


Figure 6.25: A Rough Classification of Components in Conceptual Models

Comparable viewpoints are represented as aggregations of components, according to the model presented in figure 6.25. A viewpoint is declared as an instance of the class *ConceptualModel*, and thus it can aggregate components, which express entities and/or relations (see the attributes *representsEntity* and *representRelationships* in figure 6.25).

The analysis of similarity between viewpoints results into an isomorphism between their components, reflecting their analogies. This isomorphism is a reasonable suggestion for their merging, since:

- The division of the viewpoints' components into those expressing entities and those expressing relationships is preserved by similarity analysis. This is due to the criterion of the semantic homogeneity, which precludes mappings between attributes, unless they are instances of the same attribute classes. Thus, components which are aggregated in the viewpoints by different types of attributes cannot be mapped onto each other.

- The estimation of the *classification distance* between components, detects differences regarding the properties, expressed by the classes of the described meta model.
- The estimation the *attribution distance* between viewpoints detects both the structural and the semantic differences of all the elements in their attribution graphs, since it aggregates, recursively, the overall distances between all the semantically homogeneous pairs of elements in these graphs.
- The eventual isomorphism between components of viewpoints is optimal in the sense that, it expresses the mapping with the minimum possible overall distance over all these components. Distance minimization is an objective criterion for selecting between alternative mappings and coming up with a single suggestion at the end of the viewpoints analysis stage.

6.7 Examples

In the following, we demonstrate using two examples, how similarity analysis gives isomorphisms expressing intuitive mappings between elements of different viewpoints, built according to either different or identical specification models.

Example 1. This example presents the similarity analysis of viewpoints built according to the hypothetical object-oriented data model. Both viewpoints model organizations and employees. Their main difference is the representation of the employment relationship between employees and organizations. The first of them (i.e. *ObjectOrientedModel1* in figure 6.26), expresses the employment relationship by the object attribute *OOS1IsEmployedAt*. The second viewpoint (i.e. *ObjectOrientedModel2*) expresses the same relationship by an object type (i.e. the object type *OOS2Employement*).

In addition to the structural information presented in figure 6.26, the components of the involved viewpoints are classified under the classes of the described meta model. For instance, the components *OOS1IsEmployedAt* and *OOS2Employement* have been classified as M:N, optional, separable, not homogeneous, not contemporaneous and antisymmetric binary relations. Also the components *OOS1HasCode* and *OOS2HasSecurityNumber* have been classified as identification, contemporaneous, not homogeneous, separable and existentially dependent binary relations.

Figure 6.26 also presents an intuitive mapping between the elements of these viewpoints that could be merged. Based on the structuring and the classification of these

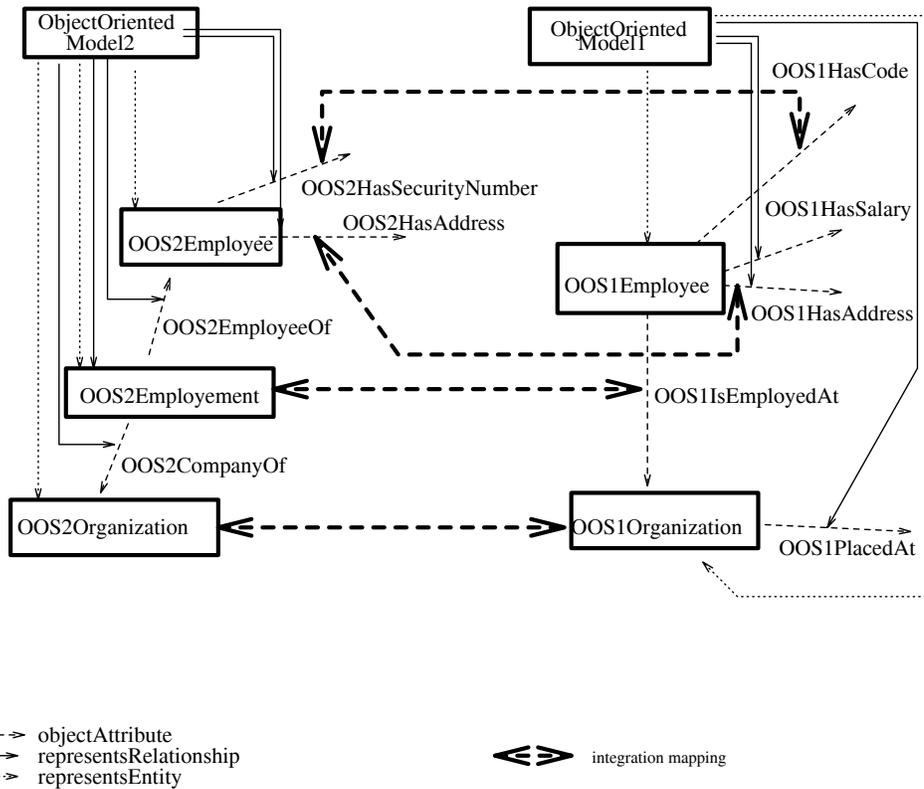


Figure 6.26: Different Viewpoints expressed in the Same Specification Model

components under the classes of the viewpoints analysis meta model, the analysis of similarity produces an isomorphism, which is presented in figure 6.27. This isomorphism includes the intuitive mapping of figure 6.26.

Notice that the components representing the organizations, the employees, their unique identifiers and the employment relations in the two viewpoints have been mapped onto each other.

In fact, the only difference between the intuitive and the estimated isomorphisms is the mapping of the attribute *OOS1HasSalary* onto the attribute *OOS2EmployeeOf*. This can be explained by noticing that if both these components had been mapped onto no element (since the rest had been already optimally mapped) the resulting isomorphism would have a larger distance with respect to the attribution of the relevant viewpoints.

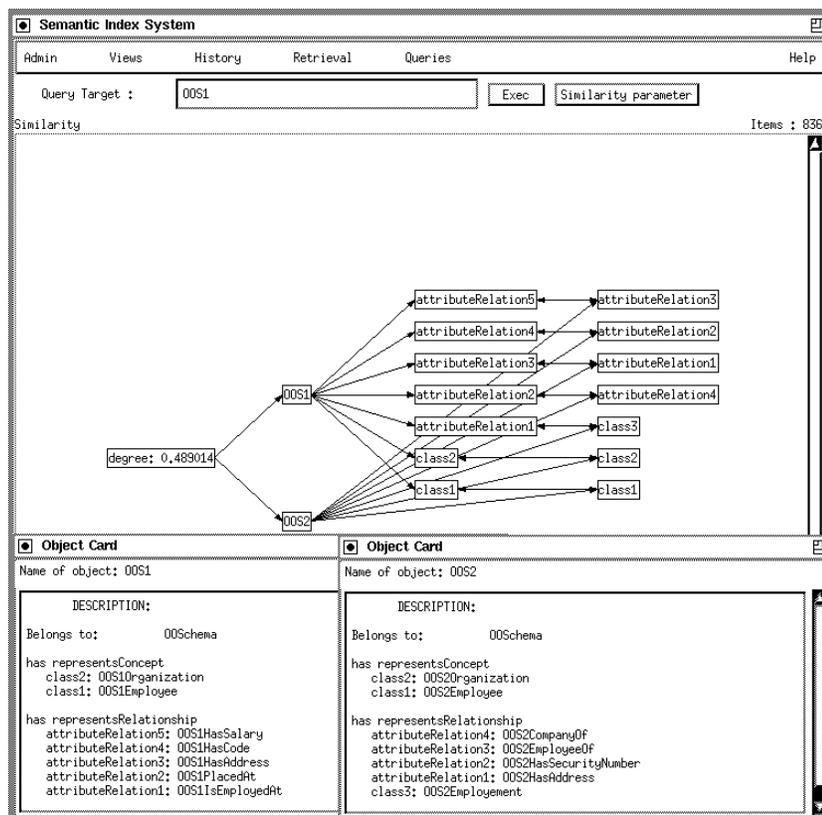


Figure 6.27: Similarity Between Object Oriented Conceptual Models

Example 2. This example taken from [HK87a], illustrates how similarity analysis can detect resemblances and discrepancies between viewpoints expressed according to different specification models. The first of them has been expressed according to the relational data model and the second according to the hypothetical object-oriented data model.

In spite of the structural discrepancies between their components, these two viewpoints express the same underlying world of persons who speak languages, live at certain addresses and travel because of their jobs. Their analogies are reflected by the correspondences shown in figure 6.28.

The analysis of their similarity results into an isomorphism identical to those correspondences, as it can be seen in figure 6.29. An interesting aspect of this isomorphism is the mapping of the isa relation between the object types *OTBusinessTraveller* and *OTPerson* onto the inclusion dependency between the fields *Pname* of the *RBusinessTraveller* and

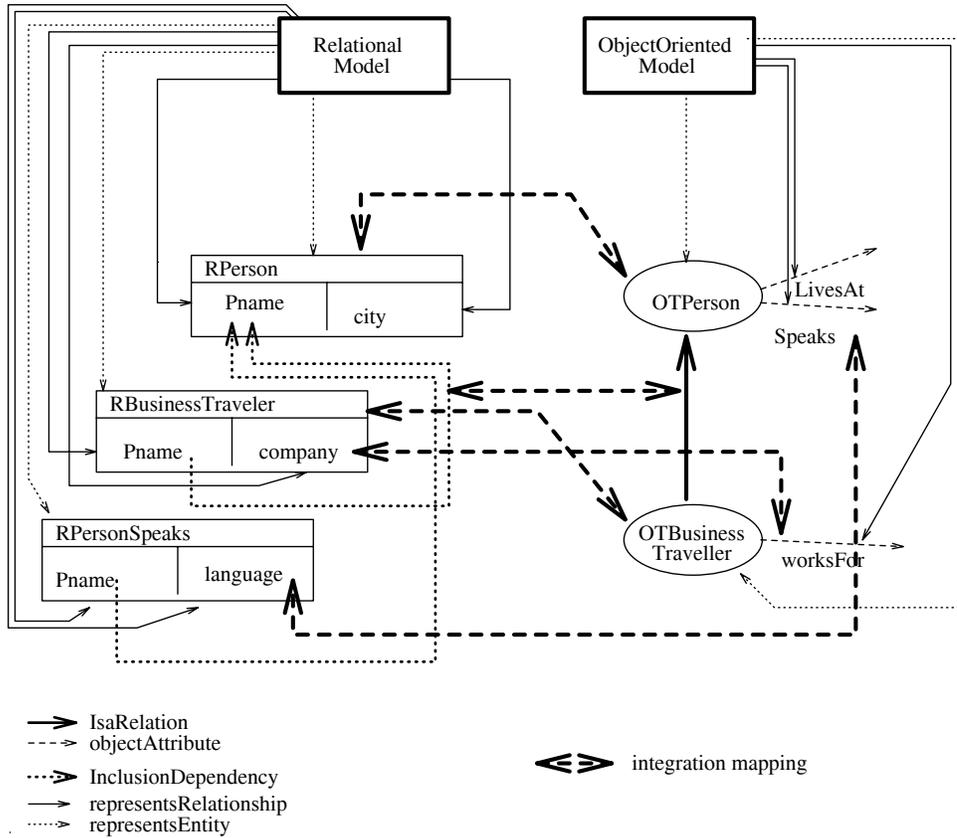


Figure 6.28: Different Viewpoints expressed in Different Specification Models

the *RPerson* relations. This mapping is derived from both the structuring of the relevant components and their classification under the appropriate classes of the meta model.

The *IsaRelation1* and the *IncDependency1* due to their instantiation under the classes *IsaRelation* and *InclusionDependency* both are instances of the class of the meta model *SetInclusionRelation*. However, this common semantics would not be sufficient for their mapping, since the *IncDependency2* shares this semantics, too. Thus, it was a candidate counterpart for the element *IsaRelation1*. The eventual selection of the *IncDependency1*, was dictated by its additional structural resemblance to the *IsaRelation1*. In fact, the element *OTBusinessTraveller*, which is the value of the *fromComponent* attribute of the *IsaRelation1* relation is more similar to the element *RBusinessTraveller*, which is the value of the same attribute of the *IncDependency1*, than the element *RPersonSpeaks*,

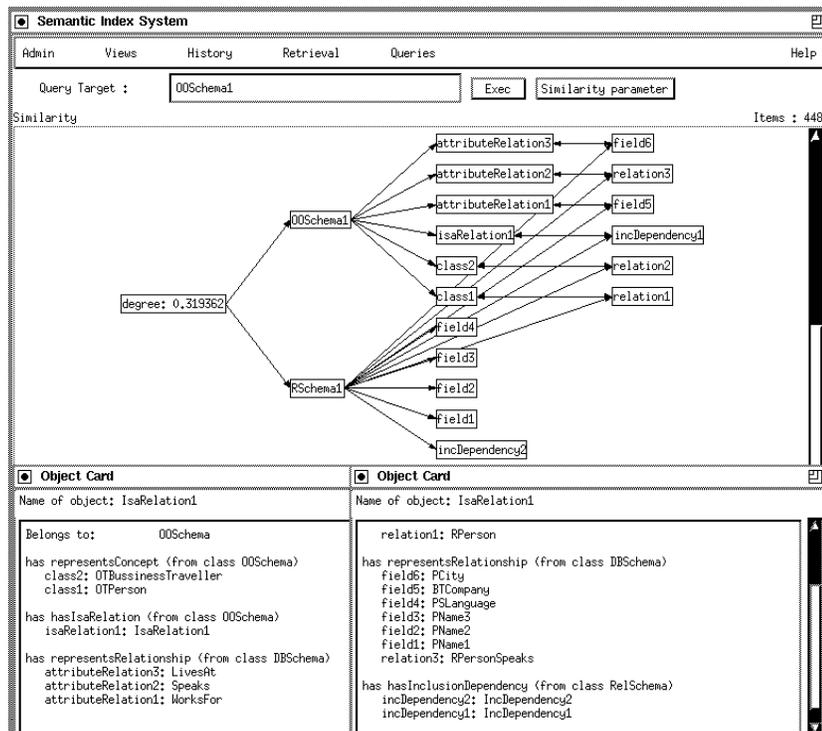


Figure 6.29: Similarity Between Object Oriented and Relational Conceptual Models

which is the value of the *fromComponent* attribute of the *IncDependency2*.

Notice also, that the eventual optimal isomorphism maps the the attribute *Speaks* on to the relation *RPersonSpeaks*. These two elements, although they are expressed through different types of constructs are semantically similar(i.e. they are both separable, not homogeneous, N:M, contemporaneous, and antisymmetric relations between persons and strings denoting languages).

Discussion. The second application of similarity analysis indicates how this model can be used as a computational reasoning mechanism for detecting discrepancies and resemblances between viewpoints. Similarity analysis is based on a meta model of domain and specification model independent semantic properties of modeling concepts, which is used to classify components in viewpoints. This analysis results into an optimal isomorphism between viewpoints, indicating the analogy between them and thus, it can be used as a plausible suggestion for merging.

Due to the domain independent characteristics of the employed meta model and the similarity analysis model, this application overcomes the domain sensitive behavior of other approaches to the problem, which use heuristics, and domain specific vocabularies to detect discrepancies and resemblances between viewpoints[LF91]. Also, as demonstrated in the second example, the whole approach is not limited by different specification models, used for expressing viewpoints, provided that the relevant viewpoint components can be classified under the appropriate meta model classes.

Various aspects of this application are still subject to further research. These include (a) an elaboration and possibly augmentation of the described meta model, with further domain and specification model independent concepts; (b) the formalization of this model; and, (c) an investigation into the sensitivity of similarity analysis to structural as opposed to semantic information about elements.

6.8 Summary

In this chapter, we demonstrated how similarity analysis can be used in modeling by reuse and integrating requirements specifications for software systems. Both tasks had inherent characteristics, dictating the appropriateness of analogical reasoning.

Two basic features of the similarity analysis model, allow its application in these tasks:

- the domain independent mechanism for elaborating analogies; and,
- the representation framework of the model.

Both features enable the analysis of specifications, expressed according to different specification models and referring to the same or different application domains.

Similarity analysis was not used to fully automate the selected tasks. Instead, it was applied to support humans by detecting analogies that could be useful in comprehending and reusing existing specifications or in suggesting which elements in two specifications are likely to express the same underlying entity, and thus they could be merged.

Chapter 7

Comparison with General Models of Analogical Reasoning

7.1 Introduction

In this chapter we compare the similarity model with other models of analogical reasoning.

Comparisons concern only general models. Purpose-driven models (i.e. models of analogical problem solving or understanding) were discussed in the second chapter while arguing about the necessity of the general models for effective interdomain analogical reasoning.

In summary, purpose-driven models cannot support effective reasoning across domains, whenever the causal knowledge, assumed to determine the salient elements of analogs, is not available or expressible through the primitives they advocate for representing it. Purpose-driven models fail to recognize that analogical reasoning may be carried out not for a specific but for a loose purpose, such as the understanding of an unfamiliar domain [THNG90, Win80,]. In such cases, even the existence of causal knowledge is uncertain [Rus88].

In the following, we compare the similarity model with the three general models of analogical reasoning, namely the Structure Mapping Theory [Gen83, Gen88a, FFG90], the Constraint Satisfaction Theory [Thag88, HT89, THNG90] and the Constrained Semantic

Transference Theory[Ind86,Ind85].

Comparisons are made with respect to particular computational and pragmatic criteria, which are introduced in advance.

7.2 A General Framework for Comparison

Our comparisons between general models and the model of analogical similarity concentrate on the following criteria:

- flexibility of elaboration;
- computational complexity;
- domain independence;
- discriminating power regarding salience;
- sensitivity to causal knowledge; and,
- required user interaction.

Flexibility of Elaboration. Flexibility concerns the ability of each model to detect as many kinds of analogy as possible. It basically depends on the mapping criteria adopted by each model. The less strict the adopted criteria, the more likely for a model to reveal analogies and thus the more the tasks it may be applicable to.

Flexibility is expected to contrast with efficiency. Flexible mapping criteria enlarge the search space of the possible mappings between the distinct elements in the descriptions of two analogs. Thus, they reduce the average ability of a model to detect analogies, without extensive computational effort. A trade-off between flexibility and efficiency is the most important criterion of assessing computational models of analogical reasoning[Thag88]. The efficiency of models is reflected by our next criterion, computational complexity.

Computational Complexity. The time and the space computational complexity is the most prominent criterion for assessing the efficiency of not only models of analogical reasoning but also of any computation.

It is important to consider both the worst and the average case complexities of the models, so as to gain a better idea of their performance.

Domain Independence. Domain independence concerns the ability of a model to detect analogies between objects belonging to different domains. Such an ability is determined by the mapping criteria employed by each model. In general, cardinality, type compatibility, structure preserving and consistency criteria are domain independent, while the knowledge preserving ones are not (see section 2.2 in chapter 2).

Discrimination of Salience. This criterion concerns the ability of a model to discriminate between elements in the descriptions of analogs having different importance to the detection of analogies.

Such an ability is a prerequisite for detecting analogies that may lead to pragmatically useful conjectures (i.e. conjectures which fulfill the purpose of performing analogical reasoning in a specific context) about the target analogs.

It is also very important to provide fine-grain distinctions between salient and non salient elements in cases where the elaboration of analogies is interleaved with the retrieval of the analogs. Empirical studies have indicated that even less salient elements may accomplish the retrieval of potentially useful source analogs and the detection of analogies, especially when such elements are predictive of more important ones, missing from the descriptions of analogs due to incompleteness [Gen88b, SH88].

Sensitivity to Causal Knowledge. This criterion regards the ability of a model to operate under different amounts of causal knowledge present in descriptions of analogs.

As discussed in chapter 2, causal knowledge is not likely to be available due to ignorance, acquisition cost or inability to express it using specific representation primitives. Thus, if a model aims to be applicable to tasks where, in spite of the absence or the limited presence of such knowledge, analogical reasoning is still required, the relative insensitivity to causal knowledge is necessary.

Sensitivity relates to the adopted mapping criteria and the estimation of salience by models of analogical reasoning. Most of the models of analogical problem solving are extreme examples of analogical reasoners, which cannot operate in tasks, where causal knowledge is totally absent. This is because such models base the entire elaboration of analogies and the estimation of salience on causal knowledge, expressed using specific representation primitives.

Required User Interaction. This criterion concerns the information from the user in advance to, or during the process of elaborating analogies.

Many models of analogical reasoning require such information, which may include qualitative or quantitative assessments of the importance of analogs' elements or distance measures between such elements. The rationale behind the acquisition of such information is to tune elaboration according to the particular characteristics of the application context.

However, the more the information required from the user, the more likely that he/she will be unable to provide it, something that may considerably reduce the operational ability of a system.

Ideally, the user should have to provide only the minimum amount of information necessary for detecting analogies, which is the description of the target analog.

7.3 Comparison with the Structure Mapping Theory

The Structure Mapping Theory has been introduced as a general model of analogical reasoning, which realizes key aspects of the elaboration of analogies by humans, indicated by relevant empirical studies[Gen83,Gen88a]. This theory has been implemented by the Structure Mapping Engine(SME, see [FFG90]).

Flexibility of Elaboration. The similarity model and SME resemble in restricting the space of all the possible mappings between the elements of two analogs, using the same cardinality criterion, isomorphic mappings. The plausibility of this cardinality criterion has been empirically verified by studies on the elaboration of analogies by humans[Gen88a]. Also, its adoption by most of the models of analogical reasoning(see table 2.2 in chapter 2) indicates that it is not considered as limiting the required flexibility of models in the detection of analogies.

However, SME adopts a considerably strict type compatibility criterion. It does not allow the mapping between elements of analogs unless they are identical. Identity is an overly restrictive criterion as even the developers of the SME admit[FFG90], which restricts the ability of SME to detect analogies.

Consider, for example, the analogy between the salary of an employee and the annual profit of a company. Both of them are taxable sources of income. Subject to the

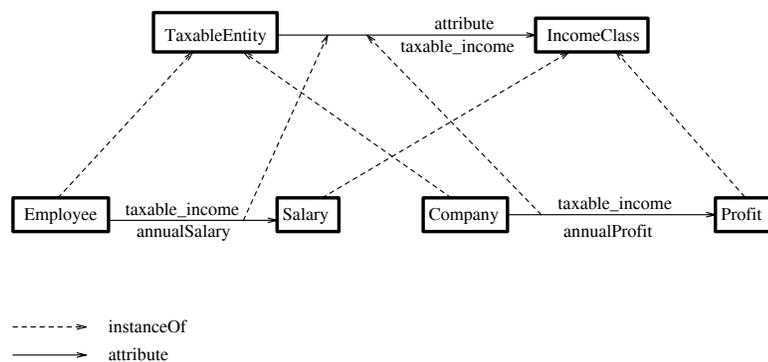


Figure 7.1: Analogical but Not Identical Relations

representation of employees, companies, salaries and annual profits, according to the conceptual model of figure 7.1, an identity based elaboration would not map the *salary* of *Employee* to the *annualProfit* of *Company*, although it should.

On the other hand, the criterion of semantic homogeneity, which is the type compatibility criterion adopted by our model of similarity, would allow this mapping, since the classification of the involved attributes under the same classes implies their semantic homogeneity.

Recall also our criticism about the inadequacy of the structure preserving criterion of SME for elaborating analogies between certain types of artifacts. In section 2.6.1 of chapter 2, we show that it is possible for configurations of artifacts in domains like software engineering to violate the structure preserving criterion, yet the relevant artifacts being analogous. Such cases indicate a limited flexibility of SME.

Hence, the similarity model is expected to be more flexible in elaboration than SME.

Computational Complexity. The current implementation the similarity model has the same worst case time complexity with SME (i.e. $O_c(N!)$, where N is the number of elements to be considered for mapping in the involved analogs). However, as shown in section 5.3.1 the expression of the estimation of the attribution distance (i.e. the distance which detects the mapping between the descriptions of two analogs in our model) as an assignment problem [PS82], makes it possible its computation in at most $2N^3$ steps. Being

implementable by an algorithm of polynomial complexity (the recursion over the transitive closures of the attribution graphs of two objects may stop at a fixed depth), our model becomes applicable to fairly large problems, unlike SME.

No comparisons can be made regarding the average time complexity, since we are not aware of such an analysis about the SME.

However, a few remarks may provide an insight on their average performance. As pointed out in [FFG90], the performance of SME depends on the very structuring of analogs. In cases, where structures of analogs are flat (i.e. they include none or only a few higher-order relations over lower-order elements), the performance of the model reaches its worst possible boundary. On the other hand, the presence of many higher-order relations restricts the search space of the different possible mappings, since mappings between elements, which are not both related by the same higher-order relations are excluded. In such cases, the performance of SME is improved substantially.

A similar result was derived analytically in section 5.3.2 for the similarity model. The average time complexity of the model depends exponentially on the average number of the attributes of analogs in each of the semantically homogeneous categories (i.e. the parameter L in formula 5.13 in chapter 5). The more such categories, the less the attributes in each of them. Thus, we can expect a faster average performance of similarity analysis whenever analogs are described according to meta-models introducing many different kinds of relations as distinct classes of attributes.

As discussed in [FFG90], it is not always convenient to interconnect lower-order attributes and relations, in descriptions of analogs, with higher-order relations, having a causal substance. On the other hand, we expect classifications of attributes and relations under common classes be easier. Thus, we believe that it will be more convenient to structure analogs in a way improving the average performance of the similarity model rather than the average performance of SME.

Domain Independence. Both models are domain independent. This is because all their common mapping criteria (i.e. the isomorphic mappings restrictions) and the non common ones (i.e. the identity and systematicity of SME and the semantic homogeneity and minimum distance isomorphism of the similarity analysis model) are not domain specific.

Discrimination of Salience. As discussed in chapter 2, SME adopts a weak criterion for discriminating between elements with different salience in the descriptions of analogs. This is because the *order* of predicates may not always reflect their importance to analogy and furthermore, it is very sensitive to different representation choices.

Unlike it, the similarity model can provide more detailed distinctions of importance due to its functions for measuring beliefs to characteristicity, abstractness and determinance of attributes.

In figure 7.2, we give an example where the similarity model is capable of discriminating between elements of the same order but different salience, that would be assessed as having equal salience by SME.

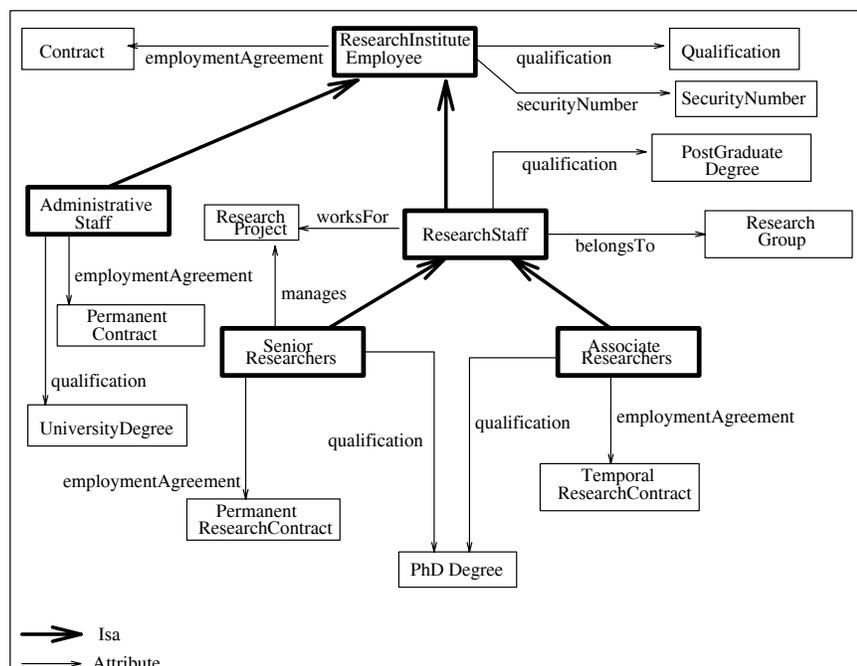


Figure 7.2: Relations of the Same Order with Different Salience

The attributes *securityNumber* and *employmentAgreement* in the taxonomy of employees of this figure, would have been represented as first-order predicates in the representation framework of SME. This is because they both relate entities instead of other relations.

Thus, they could not have been distinguished in terms of salience according to the order criterion[FFG90].

On the other hand, the similarity model is able to distinguish between them. The attribute *employmentAgreement* is more characteristic than the attribute *securityNumber*, given the presented taxonomy of employee classes. The function measuring the belief to the characteristicity of these two attributes(see definition 4.12 in chapter 4), would assign belief equal to 0.8 to *employmentAgreement* and 0.2 to *securityNumber*, reflecting that the first of these attributes distinguishes between the different kinds of employees according to the criterion of characteristicity(see definition 4.6 in chapter 4).

Sensitivity to Causal Knowledge. In SME, the guidance of elaboration by higher-order relations, taken to reflect the salient elements in an analogy makes it sensitive to their absence. Notice also, that when such relations are absent(something that may be usual regarding analogs from different domains) the space of the possible mappings between lower-order elements is augmented and SME becomes less efficient.

Furthermore, the absence of higher-order relations reduces the ability of SME to compute fine-grain *structure evaluation scores*, which enable the selection of a single best isomorphism between two analogs. This is because structure evaluation scores for mappings between lower-order predicates are estimated as fractions of the structure evaluation scores estimated for the higher-order predicates involving them.

Our similarity model is less sensitive to the absence of causal knowledge. Given its own realization of salience, even in cases where the representation of attributes over conceptual models does not allow their discrimination in terms of characteristicity, abstractness and determinance, similarity analysis is capable of detecting analogies by considering all attributes as having an equal salience, and without any performance degradations.

Required User Interaction. SME and the similarity model require users to provide only the descriptions of the target analogs. Thus, they are expected to be operational even in tasks where the users are unable to provide any further information for the elaboration of analogies (e.g. assessments of salience, hints about attributes that should be mapped).

7.4 Comparison with the Constraint Satisfaction Theory

The Constraint Satisfaction Theory has been proposed as a general theory of retrieving analogs and elaborating analogies between them, based simultaneously on syntactic, semantic and pragmatic constraints[Thag88,HT89, THNG90]. This approach of multiple constraint satisfaction is claimed to support analogical reasoning effectively both in cases where reasoning has a specific purpose and in cases where it does not.

The Constraint Satisfaction Theory has been implemented by two programs. The first supports the retrieval of source analogs, given a target, and is called ARCS(acronym standing for Analog Retrieval by Constraint Satisfaction)[THNG90]. The second supports the elaboration of mappings between analogs and is called ACME(acronym standing for Analogical Constraint Mapping Engine)[HT89].

The comparison which follows has been based on descriptions of these two programs in[HT89,THNG90].

Flexibility of Mapping. The Constraint Satisfaction Theory allows elements in analogs' descriptions to be mapped even if they are not identical. This is possible when they are associated with concepts in a lexicon, related by predefined relations, provided for expressing particular kinds of semantic similarities. These relations assign different degrees of support to mapping hypotheses between their arguments(realized as *levels of excitation* in [THNG90]). In decreasing order of their associated degrees of supporting mapping hypotheses, these relations include:

1. identity ;
 2. synonymy ;
 3. hyponymy(i.e. a relation between elements having a common generalization) ;
 4. meronymy(i.e. a relation between elements, which are both parts of the same object);
- and,
5. antonymy, which assigns a negative degree of support to the mapping hypothesis between its arguments.

The creation of mapping hypotheses between elements in descriptions of analogs due to their association with concepts in the lexicon, connected by the mentioned relations(in

taxonomies, modeled after the lexical reference system *WordNet*[MFKM88]) may prove impractical. Associations with terms denoting concepts is not always easy, since terms are usually interpreted in a context dependent way and thus they may not always be used in the same sense across domains[FLGD87].

On the other hand, similarity analysis creates mapping hypotheses based on the class identity of the elements(i.e. attributes). This is more flexible. Consider that both the conceptual schema of the classes of elements in descriptions of analogs and the classification of elements under these classes may be devised freely according to the needs of the involved application domains.

Computational Complexity. Due to its parallel implementation using connectionist networks, the Constraint Satisfaction Theory seems to outperform the similarity model concerning their time complexities. However, the worst case time complexity of this theory, measured in terms of required operation cycles of the connectionist network, cannot be analytically predicted, according to[THNG90]. Thus, our comparison is not exact.

Domain Independence. Apart from the cardinality and structure preserving mapping criteria, the Constraint Satisfaction Theory provides computational solutions to retrieval and elaboration, which are not domain independent.

In addition to the possibly context and domain dependent meaning of the concepts associated with predicates, so as to attach semantics to them, the Constraint Satisfaction Theory relies on additional predicates for acknowledging elements important to analogies(i.e. predicate IMPORTANT as well as goal and condition predicates). Such predicates may be not sufficient for expressing causal knowledge in all possible domains. For instance, the implications between conditions and goals, may be definite in some domain(e.g. physics) but probabilistic in other(e.g. medicine). Thus, such implications are not easily applicable in different domains.

These reasons make us believe that similarity analysis outperforms the Constraint Satisfaction Theory, with regard to the domain independence criterion.

Discrimination of Salience. The Constraint Satisfaction Theory is informed by the user about the elements in the descriptions of analogs, which are important to analogies and the mappings that should be presumably established in particular problem solving contexts. In addition, it recognizes problem-oriented descriptions of analogs, built around

goals and condition primitives (see our discussion about the preceding criterion), which are also characterized as important, by default.

Important elements and presumed mappings are favoured in participating in the eventual mapping, between analogs, by being assigned higher activation levels, in advance to the operation of the connectionist network, that elaborates the analogy.

This sort of salience discrimination is probably more accurate and powerful than the discrimination of salience made by similarity analysis, whenever the user is able to provide the information required by the Constraint Satisfaction Theory.

However, we expect that similarity analysis will be more effective in contexts lacking precise and well-defined goals, which normally make the user ignorant of causal aspects. In such contexts, the estimation of salience on the basis of the assumption that causal knowledge is embedded in the descriptions of analogs, and thus it can be revealed through the properties of characteristicity, abstractness and determinance may be more powerful.

Sensitivity to Causal Knowledge. Both the Constraint Satisfaction Theory and the similarity analysis model are able to elaborate analogies even when causal knowledge is not available or it cannot be syntactically detected.

In such cases, all the elements in the descriptions of analogs have the same effect to the elaboration of the eventual analogical mapping.

Required User Interaction. Although the Constraint Satisfaction Theory does not enforce the user to inform it about important elements and presumed mappings for the detection of analogies, we suspect that if the user is unable to provide such information, the ability of the theory to detect useful analogies may be considerably reduced. This suspicion is due to the lack of an alternative mechanism for detecting salient elements in descriptions of analogs in the Constraint Satisfaction Theory.

7.5 Comparison with the Constrained Semantic Transference Theory

The Constrained Semantic Transference Theory(CST) has been proposed as a formal theory for understanding metaphors[Ind86].

Metaphor understanding is viewed by the authors of this theory as a generalization of analogical reasoning. Metaphor understanding may result into any possible mapping

between two domains(i.e. analogs in our terminology). Analogical reasoning is a special kind of it, since it considers only mappings, which enable inferences for the target analog, consistent with existing knowledge about it.

The comparison between this theory and other computational models of analogical reasoning is difficult, because it has never been implemented in its full theoretical details. Only an approximation of it, referred to as *Approximate Semantic Transference*, has been computationally implemented[Ind86].

Flexibility of Elaboration. The *admissibility criterion* of CST, which precludes mappings between constants in the descriptions of two analogs, unless their types are equal, is less strict than the corresponding type compatibility criterion of similarity analysis(i.e. the semantic homogeneity). The latter requires the equality of the classes(i.e. equivalent to types) of the relations to be mapped, which in turn prerequisites the equality of the classes of the involved arguments(see axiom A.3.16 in chapter 3).

Also since CST allows N:1 mappings it is more flexible than the similarity model. However, it is difficult to predict differences flexibility due to the other mapping criteria of the two models(i.e. the *coherency* of CST and the principle of the *minimum distance isomorphism* of similarity model).

Computational Complexity. As pointed out in section 2.6.1, the formal notion of consistency underlying the coherency criterion in CST is not computable. Thus, there can be no computational complexity comparison between CST and similarity analysis.

Domain Independence. Both CST and similarity analysis are domain independent, due to their mapping criteria.

Discrimination of Saliency. CST offers no computational account for distinguishing between elements with different saliency and in this sense it is much weaker than the similarity model. Notice that even the authors of this theory realize the lack of a saliency discrimination mechanism[Ind86].

Sensitivity to Causal Knowledge. The absence of a saliency estimation mechanism make CST operational under any amount of existing causal knowledge. Thus, it does not differ from the similarity model regarding this criterion.

Required User Interaction. CST resembles the similarity model in requiring only the minimum amount of information by the user for elaborating analogies (i.e. the description of the target analog).

7.6 Summary

The presented pairwise comparisons were only qualitative and based on information about the general models, available in the literature. Thus, they cannot definitely assess the real pragmatic utility of each model, especially regarding specific applications. Nevertheless, they provided an insight into the differences of the models in elaborating analogies, from the perspective of the introduced criteria.

This insight justifies our proposal for the similarity model. Our model provides an implemented elaboration mechanism which is simultaneously domain independent, operational in the absence of causal knowledge and requires the minimum possible information from the user. These characteristics are only shared by the Constrained Semantic Transference Theory, which can only be approximated computationally.

In addition, similarity analysis is able to discriminate between elements of different salience in detail, without relying on information supplied by the user, a feature held only by SMT/SME.

Also, it must be pointed out, that the similarity model is founded on a set of well-defined distance metrics and salience measuring functions, with clear axiomatic interpretations (see chapters 3 and 4). It uses no externally asserted or a-priori existing measures while elaborating analogies unlike other models (Structure Mapping and Constraint Satisfaction Theories). Thus, it provides a clear computational solution to elaboration, whose properties are amenable to analytical assessment.

Concerning the criterion of computational complexity, our model outperforms SME but seems to be worse than the Constraint Satisfaction Theory. However, the comparison with the latter model is not certain. Furthermore, the complexity of similarity analysis could be significantly reduced by a parallel implementation (see chapter 8).

Chapter 8

Conclusions and Future Research

This thesis proposes a model for elaborating analogies based on an analysis of conceptual models of objects.

Our proposal realizes the importance of analogical reasoning as a non deductive form of reasoning, concentrates on the elaboration of analogies, which is the kernel stage of analogical reasoning and tries to overcome a number of problems of existing computational models, that limit their applicability in a broad spectrum of application domains.

In this chapter, we summarize the work reported in this thesis, emphasizing the basic aspects, the novelty and the advantages of our model, over other general models of analogical reasoning. We also discuss the limitations of this research. Finally, we point out the open research issues, which were outside its original scope, but nevertheless emerged.

8.1 Inefficiencies of Models of Analogical Reasoning

Computational models of analogical reasoning are not always efficient in supporting analogical reasoning. Their drawbacks could be summarized in reference to their classification in two broad categories, namely the purpose-driven and the general models.

Purpose-driven models are characterized by their assumption that analogical reasoning is always carried out in contexts imposing well-defined purposes (e.g. goals to be achieved, solutions to specific problems, classification of entities under a-priori known taxonomies). This assumption enable purpose-driven models elaborating analogies by focusing only on characteristics of analogs, which are causally related to imposed purposes by relations available in advance. Thus, when causal relations are not available due to an inability either to acquire or to represent them using given primitives, purpose-driven models fail. Normally, such cases arise in interdomain analogical reasoning.

General models are also weak in certain aspects. First and more important, they fail to distinguish in detail salient characteristics of analogs, which are not explicitly acknowledged as such. This is because the syntactic criteria they employ for this purpose are weak. Thus, they cannot tune elaboration so as to concentrate on salient rather than non-salient characteristics. Furthermore, they adopt criteria for hypothesizing and selecting mappings, which are overly sensitive to the exact configuration of elements in the descriptions of analogs(as with the Structure Mapping Theory[Gen83,Gen88a,FFG90]) or can only be approximated computationally(e.g. Constrained Semantic Transference Theory[Ind85,Ind86]).

8.2 The Conceptual Modeling Approach and the Similarity Model

The basic motivation of the so called conceptual modeling approach to the elaboration of analogies derives from realizing that since analogical reasoning is not always driven by well-defined purposes, the adoption of the determination approach of purpose-driven models cannot lead to a generally applicable computational solution. Thus, it is necessary to adopt the syntactic similarity approach of general models(i.e. elaboration of analogies based on syntactic criteria for analyzing descriptions of analogs), and try to revise their employed mapping and salience discrimination criteria.

We adopt this approach based on a representation framework, consisting of three semantic modeling abstractions, namely classification, generalization and attribution, which have been associated with a rich semantics in the literature [BC86,HK87,PM88,MBJK90] and inherently expose analogies¹.

¹ Other semantic modeling abstractions, such as *association* and *aggregation* have been also proposed in the literature[PM88]. Their incorporation in our model would require an investigation into the relation of their semantics to analogy and the introduction of distance metrics for them.

Based on these semantics, we introduced a set of plausible mapping criteria and operationalized them by three conceptual distance metrics for classification, generalization and attribution respectively.

These metrics, novel with respect to metrics employed by other models of analogical reasoning, carry out the elaboration of analogies. In particular, the classification distance reveals whether or not two objects have an analogous substance. The generalization distance reveals rough semantic resemblances and differences between objects. Finally, the attribution distance elaborates an analogical mapping between the attributes of the objects under comparison. Since they capture separately all the different types of relations of two analogs, these metrics enhance the ability of our model to detect analogies even in cases where conceptual models of analogs are incomplete with respect to some of these relation types.

However, it must be emphasized that the ability of our model to detect analogies between analogs depends on how precisely their conceptual descriptions express their semantics. Improper modeling deteriorates the detection of analogies and cannot be identified by similarity analysis, itself.

The metrics distinguish the importance of the individual relations they take into account. Classification and generalization distances measure importance by the inverse of the depths of classes in generalization taxonomies. Also, the attribution distance measures importance of distinct attributes as salience, which is estimated by a set of belief functions to characteristicity, abstractness and determinance of attributes.

These beliefs assess the satisfiability of three properties of attributes (charactericity, abstractness and determinance), which are claimed to indicate their importance to analogy. The novelty of this approach to the distinction of salience stems from both the employed properties of attributes and the criteria for assessing their satisfiability. These criteria concern the representation of attributes in conceptual models.

By virtue of the particular distance and salience measuring functions introduced our model is:

- domain independent;

- operational under non uniform representations of analogs(i.e. representations which are not necessarily built from the same relations);
- not sensitive to the absence of causal knowledge about the salient elements;
- capable of distinguishing salient elements in conceptual models of analogs, which are not explicitly asserted as such, using specific representation primitives; and,
- free of not use any externally provided distance or salience measures.

These characteristics make it applicable to a wide spectrum of tasks involving intra or inter domain analogical reasoning. Furthermore, its fairly general representation framework provides a good basis for integrating it with tools supporting other retrieval and reasoning mechanisms, as demonstrated by the integration of the Similarity Analyzer with the Semantic Index System[CD93] (see chapter 5). Such integrations increase the potential applicability of the model in complex tasks which can be dealt with effectively, using multiple forms of reasoning, including analogical reasoning[PGKZSB92,Allen94] (e.g. requirements engineering[RW91]).

The implemented prototype of the model and its application to the modeling of requirements by analogical reuse and viewpoints analysis(see chapter 6) provides initial in favour of these claims.

This prototype has also been used for an empirical evaluation of the consistency of the generated similarity estimates with human assessments as well as the recall and the time performance of the model.

The results of our experimentation, albeit preliminary, indicated that similarity analysis:

- does not contradict human expectations about interobject analogical similarities;
- has an acceptable recall performance; and,
- has a time performance comparable to that of other general models of analogical reasoning.

8.3 Open Issues

Open issues in the context of this work relate to the need for further empirical evaluation of the proposed model and its computational complexity.

8.3.1 Empirical Evaluation

Our claims about the scope of applicability and the effectiveness of the similarity model, have to be grounded on an extensive empirical evaluation. Such an evaluation should include tasks varying in:

- their application domains;
- the specificity of the purposes and the goals they impose; and,
- the representation models used for describing potential analogs.

The illustrative applications presented in chapter 6 were selected to cover these dimensions. However, the evidence they provided, although encouraging, is only preliminary.

Ideally, further empirical evaluation should be attempted in real application environments, involving tasks with the prescribed differences. Examples of such environments include software companies wishing to reuse artifacts of the entire software life-cycle (i.e. code, designs and specifications) by analogy and mechanical or electrical manufacturing industries, where analogical reasoning may support the very process of designing artifacts or detecting operational failures.

The integration of similarity analysis into the Semantic Index System[CD93] provides a platform suitable for such applications. In fact, the system meets a number of pragmatic data management and usability requirements, such as:

- the effective management of large collections of objects (currently tested size of the order of 10^6);
- concurrency control, transaction-based operations;
- friendly user interface; and,
- interactive data entry facilities.

8.3.2 Computational Complexity

As discussed in chapter 5, both the worst and the average case time complexity of the current implementation of similarity analysis are exponential.

In addition to reimplementing the attribution distance using an algorithm solving the assignment problem[PS82], it is necessary to investigate possible ways of reducing the computational cost of similarity analysis further. We see two basic such ways:

1. control of the recursion in estimating the attribution distance; and
2. parallel implementation of the model.

Recursion Control. Controlling recursion during the estimation of the attribution distance by an algorithm solving the assignment problem can make the complexity of the entire similarity analysis polynomial. As indicated by formula 5.5 of chapter 5, stopping recursion at the first two levels of the attribution graphs of analogs, would reduce the worst time complexity of similarity analysis to

$$B(A+1) + CA^2$$

where A and B are polynomial factors as it is evident from formulas 5.2 and 5.6, respectively.

Preliminary evidence about the gain in performance due to recursion control was presented in section 5.6.3 of chapter 5.

However, it must be pointed out that control of recursion cannot guarantee that the detected isomorphism between attributes of analogs will be optimal. Nevertheless, subject to the quality of the approximation of the optimal isomorphism, this solution to the problem of computational complexity may be satisfactory.

In general, the quality of near optimal isomorphisms depends on the correlation between the classification/generalization and the attribution distances of objects. This is because at the end of recursion the distances between the values of attributes are estimated by aggregating their classification and generalization distances(see *ObjectDistance* algorithm in section 5.2.11 of chapter 5). Initial experiments have indicated positive correlations between such distances(see table 3.1 in chapter 3). However, further experimentation is required for determining "good" control levels of recursion. This experimentation

must consider particular representation characteristics of different types of analogs.

Parallel Implementation. The similarity model is parallelizable in at least two ways.

First, the classification, generalization and attribution distances can be computed in parallel. Second, the computation of the attribution distance can be carried out in parallel, itself. As discussed in section 5.3.1, the semantically homogeneous attributes of analogs are partitioned into disjoint categories. Attributes belonging to different semantically homogeneous categories cannot be mapped onto each other. Consequently, it is possible to estimate partial optimal isomorphisms between attributes in each such category in parallel, and then, merge them into an overall optimal isomorphism, in a single step. Due to the absence of any communication needs for computing partial optimal isomorphisms, this parallelization could result in high speed-ups over sequential computations. The magnitude of such speed-ups depends on the number of the categories of semantically homogeneous attributes.

A parallel reimplementation of our model requires the redesign of the algorithms and the data structures used for the computation of the attribution distance (see chapter 5). It also requires the selection of an appropriate parallel architecture for the new algorithms. Of course, the potential of a parallel implementation must be justified by a prior study of its complexity.

8.4 Related Further Research

This thesis focused on the elaboration stage of analogical reasoning. Our focus motivated by the realization that the very construction of analogical mappings between elements of analogs, though the basis for understanding an analogy and drawing conjectures from it, had not been supported effectively by previous models of analogical reasoning.

Naturally, the next concern is to address the other stages of analogical reasoning, as well. Such an extension should be based on the same underlying conceptual modeling approach (to avoid discussed limitations of other models) aiming at:

- effective retrieval of source analogs;
- better aid to drawing conjectures and justifying them; and

- consolidation of successful episodes of analogical reasoning[Ked88,Hall89].

Based on the summarization of solutions offered to these additional stages in[Hall89], we believe that two activities mostly deserve further research effort, namely:

- the retrieval of source analogs; and,
- the drawing of conjectures.

The Retrieval of Source Analogs. Retrieval, like elaboration, need to be domain independent, not sensitive to causal knowledge and operational under non uniform representations of analogs. However, it also has to be effective: it must be performed in a fast way and obtain a high recall. Fast retrieval ensures the applicability of a model on large collections of source analogs and high recall ensures that potentially useful source analogs will not remain hidden.

Proposed general models fail to meet all these requirements. The ARCS model[THNG90] fails to meet the first three requirements(see chapters 2 and 7) while SME[FFG90] and CST[Ind86] fail to satisfy the speed concerns due to their computational complexities.

Hence, research focusing on how similarity analysis could support fast retrieval with a high recall performance is worth being undertaken. The possible reductions of complexity, described in the preceding section as well as the possible performance improvements discussed in the section 5.4 are relevant.

Furthermore, we could investigate the possibility of using only the classification and generalization distances for retrieving analogs. This would eliminate the costs of estimating the attribution distance and the salience of attributes.

Notice that, as discussed in chapter 3, the classification and the generalization distances correlate with the attribution distance(see table 3.1). Thus, during retrieval where the main problem is to choose from a large collection of source objects, those being potentially useful for analogical conjectures rather than detecting analogies between them in detail, these distances could be used as indicators of such analogies. After all objects with high classification and generalization distances are not likely to have attributes satisfying the criterion of the semantic homogeneity or attributes in common.

Drawing of Conjectures. The utilization of analogical mappings in drawing conjectures about target analogs is one of the hardest problems of analogical reasoning[Rus88,Hall89]. This is because conjectures must be consistent with existing knowledge about the target and pragmatically useful(i.e. they must fulfill the purpose of reasoning)[Grein88a,Grein88b].

As discussed in chapters 2 and 7, consistency is not decidable in the general case. Also the lack of precise purposes make impossible the automatic detection of the pragmatic utility of conjectures. Due to these difficulties, we think, together with other authors[McD79,Ash90,Aha91,Maid92], that a cooperation between human agents and systems performing analogical reasoning is necessary for drawing appropriate conjectures. The basis of such a cooperation is simple: systems of analogical reasoning must aid humans in drawing and justifying conjectures(i.e. the *cooperative approach to analogical reasoning*[Maid92]).

At its current stage of development, the prototype of the similarity model offers two sorts of information that could aid inferencing, namely:

- the analogical mapping between the attributes of analogs; and,
- the overall similarity measure indicating the aptness of detected analogies.

This aid should be enhanced. Research towards this goal could focus on the development of a process model for the inferencing activity.

This model could be activated just after the stage of elaboration so as to let humans choose between alternatives on how both the common and the non common elements of two analogs could be utilized for drawing conjectures(see section 6.2.2 in chapter 6). We believe that a process model should not force the user to take a particular action. It should only inform him about the available possibilities. It should also enable the storage of episodes of context-dependent successful and unsuccessful analogical inferences and the use of such episodes as examples or counterexamples for alternative possibilities of actions. In other words, we suggest research towards the development of a *contextual decision-oriented* process model for analogical reasoning[Ro194].

Storing episodes raises quite important questions such as:

- how to choose inference episodes, with a potential of being useful in the future; and
- whether it is worth to detect analogies between episodes so as to specialize general choices of the initial process model(i.e. the question of the adaptability of the model).

Notice that such episodes may not always be episodes of analogical problem solving as in[Car86]. Instead, they may also represent conjectures drawn for any possible purpose subject to the context of reasoning.

Glossary(English-Greek)

In the following, we translate the important terms appeared in this dissertation in the Greek language. Our translation expresses the meaning of those terms as they are used in the context of this dissertation and the literature on analogical reasoning, conceptual modeling and/or uncertainty management.

abstractness: βαθμός αφαίρεσης

abstract attribute: αφηρημένο γνώρισμα

abstract class: αφηρημένη κλάση

aggregation: συγκρότηση

analogical conjecture: αναλογικό συμπέρασμα

analogical problem solving: αναλογική επίλυση προβλημάτων

analogical reasoning: αναλογικός συλλογισμός

analogical recognition(retrieval): αναλογική ανάκτηση

analogical similarity: αναλογική ομοιότητα

aptness of analogy: ένταση της αναλογίας

aptness measure of analogy: μέτρο έντασης της αναλογίας

association: ομαδοποίηση

attribute: γνώρισμα

attribution: απόδοση γνωρισμάτων

attribution distance: απόσταση γνωρισμάτων(ολική)

basic probability assignment: συνάρτηση βασικής πιθανότητας

belief: εμπιστοσύνη

belief function: συνάρτηση εμπιστοσύνης

case-based reasoning: περιπτωσιολογικός συλλογισμός

causal knowledge: αιτιοκρατική γνώση

causal relation: αιτιοκρατική σχέση

charactericity: χαρακτηριστικότητα

characteristic attribute: χαρακτηριστικό γνώρισμα

classification: ταξινόμηση

classification distance: απόσταση ταξινόμησης

coherency: συνεκτικότητα

compositional reuse: συνθετική αναχρησιμοποίηση

conceptual model: εννοιολογικό/σημασιολογικό μοντέλο

conceptual modeling: εννοιολογική/σημασιολογική μοντελοποίηση

conceptual schema: εννοιολογικό/σημασιολογικό σχήμα(βάσεων δεδομένων)

concrete class: συγκεκριμένη κλάση

consistency preserving mapping criteria: κριτήρια απεικόνισης διασφαλίζοντα τη συνέπεια

consolidation of analogy: παγίωση αναλογίας

contemporaneous relation: συγχρονική σχέση

deductive reasoning: παραγωγικός συλλογισμός

determinance of attribute: προσδιοριστικότητα γνωρίσματος

determinative attribute: προσδιοριστικό γνώρισμα

distance function: συνάρτηση απόστασης

dominance of attribute(feature dominance): κυριαρχικότητα γνωρίσματος

elaboration of analogies: εντοπισμός αναλογιών
 extension of class: σύνολο αντικειμένων μίας κλάσης
 homogeneous relation: σχέση ομοειδών στοιχείων
 frame of discernment: σύνολο αμοιβαία αποκλειόμενων ενδεχομένων
 generalization: γενίκευση
 generalization distance: απόσταση γενίκευσης
 generalization relation(isa relation): σχέση γενίκευσης
 identification distance: απόσταση ταύτισης
 isomorphic mapping(isomorphism): ισομορφισμός
 inference episodes: περιπτώσεις εξαγωγής συμπερασμάτων
 inheritance of attribute: κληρονόμηση γνωρίσματος
 instance-of relation: σχέση συγκεκριμενοποίησης
 instantiation: συγκεκριμενοποίηση
 instantiation level: επίπεδο συγκεκριμενοποίησης
 interdomain analogies: διαπεδικές αναλογίες
 intension of object: σύνολο γνωρισμάτων ενός αντικειμένου
 intradomain analogies: ενδοπεδικές αναλογίες
 knowledge acquisition: συλλογή γνώσεων
 knowledge acquisition bottleneck: δυσλειτουργική συλλογή γνώσεων
 mapping criteria: κριτήρια αντιστοίχισης
 meronymic relation: σχέση μέρους-όλου
 meta class: μετακλάση
 modeling construct: στοιχείο κατασκευής μοντέλου
 model of uncertainty: μοντέλο αβεβαιότητας
 object identifier: στοιχείο ταυτότητας αντικειμένου
 object-oriented data model: οντοκεντρικό μοντέλο δεδομένων
 ontological uniformity: οντολογική ομοιογένεια

original class of attribute: αρχέτυπη κλάση γνωρίσματος

original domain class of attribute: αρχέτυπο πεδίο ορισμού γνωρίσματος

primitive type: πρωτογενής τύπος(αντικειμένων ή τιμών)

purpose-driven model: μοντέλο σκοπιμότητας

recall(information retrieval): βαθμός ανάκλησης

refining class: κλάση διαφοροποίησης(γνωρίσματος)

requirements engineering: τεχνολογία απαιτήσεων λογισμικού

requirements specification: διατύπωση απαιτήσεων

salience of attribute: σπουδαιότητα γνωρίσματος

scope of attribute: εύρος ισχύος γνωρίσματος

semantics: σημασία

semantic data model: σημασιολογικό μοντέλο δεδομένων

semantic homogeneity: σημασιολογική ομοιογένεια

semantic heterogeneity: σημασιολογική ετερογένεια

semantic modeling abstraction: σημασιολογική αφαίρεση

semantic overloading: δυνητική πολυσημία

separable relation: σχέση διαχωρίσιμων στοιχείων

similarity measure: μέτρο ομοιότητας

software reuse: αναχρησιμοποίηση λογισμικού

source analog: αρχικό στοιχείο της αναλογίας

specialization: εξειδίκευση

specialization depth: βάθος σε ιεραρχία εξειδίκευσης

specification model: μοντέλο διατύπωσης απαιτήσεων λογισμικού

structure preserving mapping criteria: κριτήρια αντιστοίχισης τηρούντα τη δομή

subjective probability: υποκειμενική πιθανότητα

subclass: υποκλάση

superclass: υπερκλάση

target analog: τελικό στοιχείο της αναλογίας

token: ατομικό αντικείμενο

type compatibility mapping criteria: κριτήρια αντιστοίχισης διασφαλίζοντα τη συμβατότητα του τύπου

viewpoint: υποκειμενική περιγραφή απαιτήσεων

viewpoints integration: σύνθεση υποκειμενικών περιγραφών απαιτήσεων

Bibliography

[Aga88] Agassi J., *Analogies Hard and Soft*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[AG90] Ashworth C., Goodland M., *SSADM A Practical Approach*, McGraw Hill, London, 1990

[AH87] Abiteboul S., Hall R., *IFO: A Formal Semantic Database Model*, ACM Transactions on Database Systems, 12(4), December 1987

[Aha91] Aha D., *Case-Based Learning Algorithms*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1991

[Allen94] Allen B., *Case-Based Reasoning: Business Applications*, Communications of the ACM, 33(3), 1994

[Ash90] Ashley K., *Modeling Legal Argument: Reasoning with Cases and Hypotheticals*, A Bradford Book, The MIT Press, 1990

[Bar89] Bareiss R., *Exemplar-Based Knowledge Acquisition: A Unified Approach to Concept Representation, Classification and Learning*, Academic Press Inc, 1989

[BC85] Bourrelly L., Chouraqui E., *A Formal Approach to Analogical Reasoning*,

Approximate Reasoning in Expert Systems, Elsevier Science Publishers, 1985

[BFL83] Brachman R.J., et al., *Krypton: A Functional Approach to Knowledge Representation*, IEEE Computer, Special Issue on Knowledge Representation, 16(10), 1983

[BR87] Biggerstaff T., Richter C., *Reusability Framework, Assessment, and Directions*, IEEE Software, March 1987

[BK86] Bhatnagan R.K., Kanal L., *Handling Uncertain Information: A Review of Numeric and Non Numeric Methods*, Uncertainty in Artificial Intelligence, Elsevier Science Pub., 1986

[BK87] Banerjee J., Kim W., *Semantics and Implementation of Schema Evolution in Object-Oriented Databases*, Proceedings of SIGMOD Conference, ACM, 1987

[BK88] Bareiss R., King J., *Similarity Assessment in Case-Based Reasoning*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[BM91] Bailin S.C., Moore M., *A Knowledge-Oriented Approach to Reuse*, Proceedings of the First International Workshop on Software Reusability, Dortmund, Germany, July, 1991

[BMW86] Borgida A., et al., *Generalization/Specialization as a Basis for Software Specification*, On Conceptual Modeling: Perspectives from Artificial Intelligence, Databases and Programming Languages, (ed) Mylopoulos J., Springer-Verlang, 1986

[Boeh89] Boehm B.W., *Software Risk Management*, IEEE Computer Society Press, 1989

[Bra83] Brachman R.J., *What Is-A is and isn't: An Analysis of Taxonomic Links in Semantic Networks*, IEEE Computer 15(10), 1983

[Bro84] Brodie M., *On the Development of Data Models*, On Conceptual Modeling:

Perspectives from Artificial Intelligence, Databases and Programming Languages, (eds) Brodie M., Mylopoulos J., Schmidt J., Springer-Verlag, 1984

[Bur86] Burstein M., *Concept Formation By Incremental Analogical Reasoning and Debugging*, Machine Learning: An Artificial Intelligence Approach, Vol. II, Morgan Kaufmann Publishers Inc., Los Altos, California, 1986

[Bur88] Burstein M., *Analogy versus CBR: The Purpose of Mapping*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[Cam86] Cameron J.R., *An Overview of JSD*, IEEE Transactions on Software Engineering, 12(2), 1986

[Car83] Carbonell J.G., *Learning by Analogy: Formulating and Generalizing Plans from Past Experience*, Machine Learning: An Artificial Intelligence Approach, Vol. I, Morgan Kaufmann Publishers Inc., Los Altos, California, 1983

[Car86] Carbonell J.G., *Derivational Analogy: A Theory of Reconstructive Problem Solving and Expertise Acquisition*, Machine Learning: An Artificial Intelligence Approach, Vol. II, Morgan Kaufmann Publishers Inc., Los Altos, California, 1986

[CCD92] Christoforaki M., et al., *CLIO: An Object Oriented Model of Cultural Data - Part I: General Concepts*, MUIS92.1, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, 1992

[CD93] Constantopoulos P., Doerr M., *The Semantic Index System: A Brief Presentation*, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, 1993

[CDP93] Constantopoulos P., Doerr M., Petra E., *On Classification of Object-Oriented Software for Reuse*, Proceedings of ERCIM EDRG Workshop 4, Crete, Greece, 1993

[CDV93] Constantopoulos P., Doerr M., Vassiliou Y., *Repositories for Software Reuse:*

The Software Information Base, Proceedings of the IFIP Conference on the Software Development Process, Como, Italy, 1993

[CFR91] Callan J., Fawcett T., Rissland E., *CABOT: An Adaptive Approach to Case Based Search*, Proceedings of the International Joint Conference on Artificial Intelligence, 1991

[Codd70] Codd F., *A Relational Model of Data for Large Shared Data Banks*, Communications of the ACM, 13(6), June, 1970

[Codd79] Codd F., *Extending the Database Relational Model to Capture More Meaning*, ACM Transactions on Database Systems, 4(4), December, 1979

[Cook91] Cook D., *The Base Selection Task in Analogical Planning*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1991

[CPS91] Cain T., et al., *Using Domain Knowledge to Influence Similarity Judgements*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1991

[DD92] Dadouris C., Doerr M., *The Programmatic Query Interface for the SIB, User's Manual*, ITHACA.FORTH.92.E2.#8, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, 1992

[DHN76] Duda E., et al., *Subjective Bayesian Methods for a Rule Based Inference System*, Proceedings of the National Computer Conference, 45, 1976

[Diaz93] Diaz R.P., *Status Report: Software Reusability*, IEEE Software, May 1993

[DK93] Doerr M., Klimathianakis P., *A TELOS Model for C++ Static Analysis Data*, Technical Report, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, September 1993

[Drake67] Drake A., *Fundamentals of Applied Probability Theory*, McGraw-Hill Book

Company, 1967

[FG91] Fertig S., Gelernter D., *FGP: A Virtual Machine for Acquiring Knowledge from Cases*, Proceedings of the International Joint Conference on Artificial Intelligence, 1991

[FFG90] Falkenhainer B., Forbus K., Gentner D., *The Structure Mapping Engine: Algorithm and Examples*, Artificial Intelligence, 41, 1990

[FK85] Fikes R., Kehler T., *The Role of Frame-Based Representation in Reasoning*, Communications of the ACM, 28(9), 1985

[FKN91] Fankhauser R., Kracker M., Neuhold E., *Semantic vs. Structural Resemblance of Classes*, SIGMOD Record 20(4), 1991

[FLGD87] Furnas G.W., et al., *The Vocabulary Problem in Human-System Communication*, Communications of the ACM, 30(11), 1987

[Fre83] Freeman P., *Reusable Software Engineering: Concepts and Research Directions*, Proceedings of ITT Workshop on Reusability in Programming, September, 1983

[Fug93] Fuggetta A., *A Classification of CASE Technology*, IEEE Computer, December 1993

[GB91] Gangopadhyay D., Barsalou T., *On the Semantic Equivalence of Heterogeneous Populations in Multimodel, Multidatabase Systems*, SIGMOD Record 20(4), 1991

[Gen83] Gentner D., *Structure Mapping: A Theoretical Framework for Analogy*, Cognitive Science, 7, 1983

[Gen88a] Gentner D., *Analogical Inference and Analogical Access*, Analogica, (ed) Armand Prieditis, Morgan Kaufmann Pub., 1988

[Gen88b] Gentner D., *Finding the Needle: Accessing and Reasoning From Prior Cases*,

Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[Gil87] Giles J.R., *Introduction to the Analysis of Metric Spaces*, Australian Mathematical Society, Cambridge University Press, 1987

[GL85] Gentner D., Landers R., *Analogical Reminding: A Good Match is Hard to Find*, Proceedings of International Conference on Systems, Man and Cybernetics, Tucson, AZ, 1985

[Gre84] Greenspan S., *Requirements Modeling: A Knowledge Representation Approach to Software Requirements Definition*, Ph.D thesis, Dept. of Computer Science, University of Toronto, 1984

[Grein88a] Greiner R., *Learning and Understanding by Analogies*, Artificial Intelligence, 35, 1988

[Grein88b] Greiner R., *Abstraction Based Analogical Inference*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[Gro78] Grossberg S., *A Theory of Visual Coding, Memory and Development*, Formal Theories of Visual Perception, (eds) Leeuwenberg E., Buffart J., Wiley, New York, 1978

[GT86] Gentner D., Toupin C., *Systematicity and Surface Similarity in the Development of Analogy*, Cognitive Science, 1986

[GW61] Glasser G. J., Winter R.F., *Critical Values of the Coefficient of Rank Correlation for Testing the Hypothesis of Independence*, Biometrika, 48, 1961

[Hal93] Halkia P., *Implementation Description of the Set Manipulation Mechanism for the Semantic Index System*, CWP no. 7, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, 1993

[Hall89] Hall R., *Computational Approaches to Analogical Reasoning: A Comparative Analysis*, Artificial Intelligence, 39, 1989

[Hec86] Heckerman D., *Probabilistic Interpretation of the MYCIN's Certainty Factors*, Uncertainty in Artificial Intelligence, Elsevier Science Pub., 1986

[HK87a] Hull R., King R., *Semantic Database Modeling: Survey, Applications and Research Issues*, ACM Computing Surveys, 19(3), 1987

[HK87b] Holyoak K., Koh K., *Surface and Structural Similarity in Analogical Transfer*, Memory and Cognition 15(4), 1987

[Hof93] Hofmann H., *Requirements Engineering: A Survey of Methods and Tools*, Institute for Informatics, University of Zurich, March 1993

[HT89] Holyoak K., Thagard P., *Analogical Mapping by Constraint Satisfaction*, Cognitive Science, 13, 1989

[Ind85] Indurkha B., *A Computational Theory of Metaphor Comprehension and Analogical Reasoning*, Ph.D Thesis, Dept. of Computer and Information Science, University of Massachusetts, Amherst, Mass, 1985

[Ind86] Indurkha B., *Constrained Semantic Transference: A Formal Theory of Metaphors*, Synthese, 68(3), Reidel Publishing Company, 1986

[JFH92] Johnson W.L., et al., *Representation and Presentation of Requirements Knowledge*, IEEE Transactions on Software Engineering, 18(10), 1992

[JMSV92] Jarke M., et al., *DAIDA-An Environment for Evolving Information Systems*, ACM Transactions on Information Systems, 10(1), 1992

[Joh88] Johnson R., *Designing Reusable Classes*, Journal of Object Oriented Programming, June/July 1988

- [Kat94] Katalagarianos P., *Employing Genericity and Case-Based Reasoning to Effectively Reuse Code*, Ph.D Thesis, Dept. of Computer Science, University of Crete, February, 1994
- [Ked85] Kedar-Cabelli S., *Purpose-Directed Analogy*, Proceedings of the 7th Annual Conference of the Cognitive Science Society, Irvine, CA, 1985
- [Ked88] Kedar-Cabelli S., *Analogy-From a Unified Perspective*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988
- [KC86] Khoshafian S., Copeland G., *Object Identity*, Proceedings of the International Conference on Object-Oriented Programming, Systems and Languages(OOPSLA '86), September, 1986
- [Keen92] Keen M., *Presenting Results of Experimental Retrieval Comparisons*, Information Processing & Management, 28(4), 1992
- [Kev84] Kevork K., *Analysis of Regression and Correlation*, Statistical Methods, Vol. III, Economical University of Athens, Athens, 1984
- [Kol84] Kolonder J., *Retrieval and Organizational Strategies in Conceptual Memory: A Computer Model*, Hillsdale, NJ: Lawrence Erlbaum Associates, 1984
- [Kow65] Kowalski J., *Topological Spaces*, Academic Press, N.Y, 1965
- [KL89] Kim W., Lochovsky F., *Object-Oriented Concepts, Databases, and Applications*, (eds) Kim W., Lochovsky F., ACM Press, 1989
- [KMSB89] Koubarakis M., et al., *Telos: Features and Formalization*, Technical Report, Technical Report Series, FORTH/CSI/TR/1989/018, Institute of Computer Science, Foundation for Research and Technology-Hellas, Iraklion, Crete, Greece, 1989

[Kol88] Kolonder J., *Judging Which is the "Best" Case for a Case-Based Reasoner*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[Kru92] Krueger C., *Software Reuse*, ACM Computing Surveys, 24(2), June 1992

[KSS85] Kolonder J.L., et al., *A Process Model of Case-Based Reasoning in Problem Solving*, Proceedings of International Joint Conference on Artificial Intelligence(IJCAI-85), Los Angeles, CA, 1985

[Kuip88] Kuiper T., *Inductive Analogy By Similarity and Proximity*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[LF91] Leite J, Freeman P., *Requirements Validation Through Viewpoint Resolution*, IEEE Transactions on Software Engineering 17(12), 1991

[LGN81] Lundeberg M. et al., *Information Systems Development: A Systematic Approach*, Prentice Hall, Englewood Cliffs, 1981

[LH86] Lubars M., Harandi M., *Intelligent Support for Software Specification and Design*, IEEE Expert, Winter 1986

[LRV88] Lecluse C., et al., *O₂, an Object-Oriented Data Model*, Proceedings of SIGMOD, ACM, 1988

[Maid92] Maiden N., *Analogical Specification Reuse During Requirements Analysis*, Ph.D Thesis, Dept. of Business Computing, City University, London , July 1992

[Mal92] Malcolm E., *SSADM Version 4: A User's Guide*, McGraw Hill, 1992

[MBJK90] Mylopoulos J., et. al., *Telos: Representing Knowledge About Information Systems*, ACM Transactions on Information Systems, 8(4), 1990

[McD79] McDermott J., *Learning to Use Analogies*, Proceedings of the International Joint Conference on Artificial Intelligence(IJCAI-79), Tokyo, 1979

[Mey89] Meyer B., *Object Oriented Software Construction*, C.A.R Hoare, Series Editor, 1989

[MFKM88] Miller A., et al., *WORDNET: An Electronic Lexical Reference System Based on Theories of Lexical Memory*, Revue Quebecoise Linguistique, 17, 1988

[MH91a] Miriyala k., Harandi M., *The role of analogy in Specification Derivation*, Proceedings of the 6th Annual Conference on Knowledge-Based Software Engineering, IEEE Computer Society Press, 1991

[MH91b] Miriyala k., Harandi M., *Automatic Derivation of Formal Software Specifications From Informal Descriptions*, IEEE Transactions on Software Engineering, 17(10), October 1991

[ML84] Mylopoulos J., Levesque H., *An Overview of Knowledge Representation, On Conceptual Modeling: Perspectives From Artificial Intelligence, Databases and Programming Languages*, (eds) Brodie M., Mylopoulos J., and Schmidt J., Springer-Verlang, 1984

[Mos89] Mostow J., *Design By Derivational Analogy: Issues in the Automatic Replay of Design Plans*, Artificial Intelligence, 40, 1989

[MP93] Motschnig-Pitrik R., *The Semantics of Parts vs. Aggregates in Data Knowledge Modeling*, Proceedings of the 5th International Conference on Advanced Information Systems Engineering, (CAiSE '93), (eds) Rolland C., Cauvet C., LNCS 685, June 1993

[MR91] Mylopoulos J., Rose T., *Case-Based Reuse for Information System Development: The Techne Project*, Proceedings of the First International Workshop on Software Reusability, Dortmund, Germany, July 1991

- [MS91] Maiden N., Sutcliffe A., *Analogical Matching for Specification Reuse*, Proceedings of the 6th Annual Conference on Knowledge-Based Software Engineering, IEEE Computer Society Press, 1991
- [MS92] Maiden N., Sutcliffe A., *Exploiting Reusable Specifications through Analogy*, Communications of the ACM, 35(4), 1992
- [MS93] Maiden N., Sutcliffe A., *Architecture & Algorithms of the Domain Matcher*, NATURE CU-93-OOD, Dept. of Business Computing, City University, London, 1993
- [MSOP86] Maier D., et al., *Development of an Object-Oriented DBMS*, Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, (OOPSLA '86), Portland, Oregon, September 1986
- [Mun81] Munyer C., *Analogy as a Means of Discovery in Problem Solving and Learning*, Ph.D Thesis, University of California at Santa Cruz, CA, 1981
- [Myl90] Mylopoulos J., *Object-Oriented Knowledge Representation*, Object-Oriented Databases: Analysis, Design and Construction, (eds) Meersman R., Kent W., North-Holland, 1990
- [NA90] Ng K., Abramson B., *Uncertainty Management in Expert Systems*, IEEE Expert, April 1990
- [Niin88] Niiniluoto I., *Analogy and Similarity in Scientific Reasoning*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988
- [NKF93] Nuseibeh B., et al., *Expressing the Relationship between Multiple Views in Requirements Specifications*, Proceedings of International Conference on Software Engineering, (ICSE-15), 1993
- [Nov88] Novick L., *Analogical Transfer: Processes and Individual Differences*,

Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[PBH90] Porter W.B., et al., *Concept Learning and Heuristic Classification in Weak-Theory Domains*, Artificial Intelligence, 45, 1990

[PGKZSB92] Pearcee M., et al., *Case-Based Design Support: A Case Study of Architectural Design*, IEEE Expert, 7(5), October 1992

[Ple90] Plexousakis D., *An Ontology and a Possible-Worlds Semantics for Telos*, Technical Report on Knowledge Representation and Reasoning, Dept. of Computer Science, University of Toronto, KRR-TR-90-7, September, 1990

[PM88] Peckham J., Maryanski F., *Semantic Data Models*, ACM Computing Surveys, 20(3), 1988

[Pohl93] Pohl K., *The Three Dimensions of Requirements Engineering*, Proceedings of the 5th International Conference on Advanced Information Systems Engineering, (CAiSE '93), (eds) Rolland C., Cauvet C., LNCS 685, June 1993

[PS82] Papadimitriou C., Steiglitz K., *Combinatorial Optimization: Algorithms and Complexity*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1982

[Quin83] Quinlan R., *INFERNO: A Cautious Approach to Uncertain Inference*, The Computer Journal, 26(3), 1983

[RMGJB76] Rosch E., et al., *Basic Objects in Natural Categories*, Cognitive Psychology, 8, 1976

[Rol94] Rolland C., *A Contextual Approach for the Requirements Engineering Process*, Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering, SEKE '94, Printed by Knowledge Systems Institute, Skokie, 1994

- [Ross88] Ross B., *Some Psychological Results in CBR*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988
- [Rus88] Russel S., *Analogy By Similarity*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988
- [RW91] Reubenstein H., Waters R., *The Requirements Apprentice: Automated Assistant for Requirements Acquisition*, IEEE Transactions on Software Engineering, 17(3), March 1991
- [SB75] Shortliffe H., Buchanan G., *A Model of Inexact Reasoning in Medicine*, Mathematical Biosciences 23, 1975
- [SC93] Spanoudakis G., Constantopoulos P., *Similarity for Analogical Software Reuse: A Conceptual Modeling Approach*, Proceedings of the 5th International Conference on Advanced Information Systems Engineering, (CAiSE '93), (eds) Rolland C., Cauvet C., LNCS 685, June 1993
- [SC94a] Spanoudakis G., Constantopoulos P., *Measuring Similarity Between Software Artifacts*, Proceedings of the 6th International Conference on Software Engineering and Knowledge Engineering, SEKE '94, Printed by Knowledge Systems Institute, Skokie, 1994
- [SC94b] Spanoudakis G., Constantopoulos P., *Similarity for Analogical Software Reuse: A Computational Model*, Proceedings of the 11th European Conference on Artificial Intelligence(ECAI '94), Amsterdam, The Netherlands, August 1994
- [SC94c] Spanoudakis G., Constantopoulos P., *On Evidential Feature Saliency*, Proceedings of the 5th International Conference on Database and Expert Systems Applications(DEXA'94), Athens, September 1994
- [Sci89] Sciore E., *Object Specialization*, ACM Transactions on Office Information

Systems, 7(2), 1989

[Seif88] Seifert C., *Analogy and Case-Based Reasoning*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[SH88] Seifert C., Hammond C., *Why There is No Analogical Transfer*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[Sha75] Shafer G., *A Mathematical Theory of Evidence*, Princeton University Press, 1975

[She84] Shepard R., *Similarity and a law of universal generalization*, Annual Meeting of Psychonomic Society, San Antonio, Texas, 1984

[Sim85] Simpson R.L., *A Computer Model of Case-Based Reasoning in Problem Solving: An Investigation in the Domain of Dispute Mediation*, Ph.D Thesis, Technical Report GIT-ICS-85/18, Georgia Institute of Technology, Atlanta, GA, 1985

[Sjo72] Sjoberg L., *A Cognitive Theory of Similarity*, Goteborg Psychological Reports, Number 10, Vol. II, 1972

[SM91] Sutcliffe A., Maiden N., *Specification Reuse by Analogy*, Proceedings of 2nd Workshop on Next Generation of CASE Tools, (eds) Tehvanainen V-P., Lyytinen K., IOS Press, Trondheim, Norway, 1991

[SM92] Sutcliffe A., Maiden N., *Analyzing the Novice Analyst: Cognitive Models in Software Engineering*, International Journal on Man-Machine Studies, 36, Academic Press Ltd., 1992

[Smi89] Smith E.E., *Concepts and Induction*, Foundations of Cognitive Science, A Bradford Book, The MIT Press, 1989

[SN91] Sycara K., Navinhandria D., *Influences: A Thematic Abstraction for Creative Use of Multiple cases*, Proceedings of the DARPA Workshop on Case-Based Reasoning,

1991

[Som88] van Someren M.W., *Using Attribute Dependencies for Rule Learning*, European Workshop on Knowledge Representation and Organization in Machine Learning, Springer-Verlag, 1988

[Spa94a] Spanoudakis G., *Similarity Analyzer: An Implementation Overview*, Working Paper #11, Institute of Computer Science, Foundation for Research and Technology - Hellas, Iraklion, Crete, Greece, September 1994

[Spa94b] Spanoudakis G., *Similarity Analysis: An Empirical Evaluation*, Working Paper #12, Institute of Computer Science, Foundation for Research and Technology - Hellas, Iraklion, Crete, Greece, September 1994

[Spiv90] Spivey J.M., *An Introduction to Z and Formal Specifications*, Software Engineering Journal, 4(1), 1990

[Sta86] Stanley M., *CML: A Knowledge Representation Language with Application to Requirements Modeling*, Master's thesis, Dept. of Computer Science, University of Toronto, 1986

[Sto93] Storey V., *Understanding Semantic Relations*, VLDB Journal 3, 1993

[Str87] Stroustrup B., *The C++ Programming Language*, Addison-Wesley, Reading, Mass., 1987

[Su92] Su L., *Evaluation Measures for Interactive Information Retrieval*, Information Processing and Management, 28(4), 1992

[TB91] Tausend B., Bell S., *Analogical Reasoning for Logic Programming*, Proceedings of European Working Session on Learning, (ed) Kodratoff Y., Springer-Verlag, 1991

[Thag88] Thagard P., *Dimensions of Analogy*, Analogical Reasoning: Perspectives of

Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[THNG90] Thagard P. et al., *Analog Retrieval by Constraint Satisfaction*, Artificial Intelligence, 46, 1990

[TS93] Thieme C., Siebes A., *Schema Integration in Object Oriented Databases*, Proceedings of the 5th International Conference on Advanced Information Systems Engineering, (CAiSE '93), (eds) Rolland C., Cauvet C., LNCS 685, June 1993

[Tr90] Tracz W., *Where Does Reuse Start ?*, ACM SIGSOFT Software Engineering Notes, 15(2), April 1990

[Tur88] Turner M., *Categories and Analogies*, Analogical Reasoning: Perspectives of Artificial Intelligence, Cognitive Science and Philosophy, (ed) Helman D.H., Kluwer Academic Pub., Dordrecht, The Netherlands, 1988

[Tve77] Tversky A., *Features of Similarity*, Psychological Review, 44(4), July 1977

[YK80] Yamane T., Kinti A., *Mathematics for Economists*, Vol. I and II, Gutenberg, Athens, 1980

[VC91a] Veloso M., Carbonell J., *Learning by Analogical Replay in PRODIGY: First Results*, Proceedings of European Working Session on Machine Learning, (ed) Kodratoff Y., Springer-Verlang, 1991

[VC91b] Veloso M., Carbonell J., *Variable-Precision Case Retrieval in Analogical Problem Solving*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1991

[Web92] Weber M., *Implementation Models for the SIB*, ITHACA.SNI.92.E2.#2, Siemens-Nixdorf, September, 1992

[Weg87] Wegner P., *Dimensions of Object Based Language Design*, Proceedings of the ACM Conference on Object-Oriented Programming, Systems, Languages and Applications, (OOPSLA '87), 1987

[Win80] Winston P., *Learning and Reasoning By Analogy*, Communications of the ACM 23(12), 1980

[WWK88] Whitaker L., et al., *Using Qualitative or Multi-Attribute Similarity to Retrieve Useful Cases from a Case Base*, Proceedings of the DARPA Workshop on Case-Based Reasoning, 1988

[Zad86] Zadeh L., *Is Probability Sufficient for Dealing with Uncertainty in AI: A Negative View*, Uncertainty in Artificial Intelligence, Elsevier Science Publishers, 1986

[Zad88] Zadeh L., *Fuzzy Logic*, IEEE Computer, April 1988