MULTIFACETED ONTOLOGICAL NETWORKS:
METHODOLOGICAL STUDIES TOWARD
FORMAL KNOWLEDGE REPRESENTATION

Approved by:

Dr. John R. Sullins

Dr. Weidong Chen

Dr. Murat M. Tanik

MULTIFACETED ONTOLOGICAL NETWORKS: METHODOLOGICAL STUDIES TOWARD FORMAL KNOWLEDGE REPRESENTATION

A Thesis Presented to the Graduate Faculty of the
School of Engineering and Applied Science
Southern Methodist University
in
Partial Fulfillment of the Requirements
for the degree of
Master of Science
with a
Major in Computer Science
by

Mehmet M. Kayaalp
(M.D., University of Istanbul)

December 18, 1993

ACKNOWLEDGMENTS

This work would not have been possible without my advisor Dr. John Sullins and his continuous support. He has not only spent many hours to make this work as complete as it is but also built a comfortable atmosphere in which I could work as creatively and productively as I ever have been able to. I also am very grateful to Dr. Murat Tanik who brought me to SMU and has always been one of the most prominent people in my studies here. Through his efforts, I obtained my background in Computer Science and Software Engineering. Lastly, I would like to acknowledge the kind efforts of my friends, Coskun Bayrak, Mario Nascimento, Ted Pedersen and Clenio Salviano who reviewed first few chapters of this work and gave me a hard time convincing them of conceptual definitions such as information and knowledge as well as the need for frames in the Ontological Networks. All those have been precious feedbacks to me.

Kayaalp, Mehmet M. M.D., University of Istanbul, 1989

Multifaceted Ontological Networks:
Methodological Studies toward
Formal Knowledge Representation

Advisor: Assistant Professor John R. Sullins

Master of Science degree conferred December 18, 1993

Thesis completed August 30, 1993

Keywords: Knowledge representation, semantic networks, frame based representation, scientific ontology, knowledge sharing, semantic problem, information theory, knowledge theory, multifaceted modeling, biomedical modeling, artificial intelligence in medicine, ontological engineering.

Today's professionals find it more and more difficult to cope with the rapid increase in available information in their areas. In addition, it is becoming more difficult for them to communicate with other experts who might be able to interpret this information, due to the differences in terminology and underlying conceptual models between fields. To solve this problem, we propose a representation system called Multifaceted Ontological Networks into which these different perspectives can be integrated.

This methodology isolates sharable and reusable knowledge modules by extracting fundamental relationships between entities. Complex systems are decomposed in both structural and class (object oriented) terms, and different conceptual models are unified by grounding them in established areas such as natural sciences. The resulting network would contain only the factual knowledge behind these models, which

would be organized based on conceptual dependencies with minimal ontological commitments.

In order to cope with the enormous amount of information such a network would contain, users may interact with it through facets which match their own conceptual model of the world. These facets provide knowledge at the desired levels of aggregation, abstraction, and precision for the user, and in the appropriate terminology. Conflicting conceptual models can also be represented, by isolating the inconsistent knowledge in individual facets while sharing the consistent knowledge.

Conceptually, the ontological network is a combination of semantic networks (for representing and viewing relationships between entities) and frames (for representing and viewing values of properties of individual entities), while the actual implementation is in logic programming. A prototype implementation of this system has been developed for the Citric Acid Cycle, a major biochemical process in human metabolism.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVIATIONS

ALU	Arithmetic Logic Unit
AI	Artificial Intelligence
DNA	Deoxyribonucleic Acid
e.g.	for example
i.e.	in other words
medKAT	medical Knowledge Acquisition Tool
OSA	Ontological Systems Analysis
OSAS	Ontological Systems Analysis and Synthesis
OSS	Ontological Systems Synthesis
q.v.	see, or refer to
RNA	Ribonucleic Acid

GLOSSARY

Abstraction. A particular perspective or part (or model) of a system. Also, a simplification method in mathematical modeling, where *some* components are represented while others are neglected for practicality.

Aggregation. Identifying systems of relations in a single entity; giving a name to a system of concepts. Also, a simplification method in mathematical modeling, where all components are represented as a conjugated whole.

Artificial intelligence. A discipline as an integration of computer science, cognitive science, linguistics, logic, philosophy, and engineering. It deals with knowledge acquisition, representation, and reasoning in any possible ways.

Concept. An entity represented on a particular platform.

Conceptual universe. *All* entities along with their qualities and interrelations modeled in an agent's mind (mental model) or represented in a computer system.

Data. A specific type of knowledge. It describes the numerical *quality (q.v.)* of an entity.

Decoding. Receiving a piece of information and transforming it into another representation. There are certain levels of decoding. (i) Sensing (receptor or stimulus level of decoding), (ii) understanding or identification (conscious level of decoding), (iii) conceptualization or deeper understanding (knowledge level of decoding, being able to incorporate the decoded semantics with the existing knowledge).

DNA Deoxyribonucleic acid. It is a double helix string of nucleotides of which bases stores the genetic knowledge in a specific type of information that is decoded (deciphered) in a unique way throughout all species.

Entity. A real being (i.e., a thing) in nature or relation between things in time (i.e., a process); a component of knowledge or a *knowledge module (q.v.)*. An entity may also embody abstract concepts; however, they are not within the scope of this work.

Epistemology. A part of philosophy concerned with every aspect of knowledge.

Facet. An abstraction of a *conceptual universe (q.v.)*, or an interface between an agent (human or a software system) and the Multifaceted Ontological Representation of a conceptual universe *(q.v. multifaceted ontological network)*.

Formal knowledge. Unambiguous, rigorous knowledge, e.g., knowledge represented in *Ontological Networks (q.v.)* or genetic knowledge. This is in contrast to *informal knowledge (q.v.)*. It is fully connected with every relevant entity within a particular *conceptual universe (q.v.)*.

Formal knowledge representation. A methodology for representing *formal knowledge* in computers so that the information (the representation) can always be interpreted into a unique semantics that is unambiguous and rigorous for computers as well as for human beings to process.

Heuristic. Though there is no agreement on semantics of this word in general, in this work it is used as an entity or a quality describing the efficiency of the problem solving ability of human being, and the simulation of these skills within artificial intelligence programs. For a better treatment of this word please refer to (Barr and Feigenbaum 1981) pp. 28-31.

Idealization. A simplification method in mathematical modeling, where the data is represented in less precision format for practical purposes. It is also known as approximation.

Informal knowledge. Knowledge that is unstructured, not well defined and not fully connected with all existing relevant concepts in *conceptual universe (q.v.)* of an agent. The agent who owns the informal knowledge cannot be certain about its conceptual content; i.e., he may not be sure about all subconcepts involved, relations between them, relations between them and other concepts, or relations between the main concept and other concepts. Person A

may understand informal knowledge to some extent and can utilize it practically. He might, however, face difficulty communicating with person B who has conceptualized that knowledge in terms of relations with *other* concepts and subconcepts of which person A is not fully aware.

Information. A code created by altering physical structure and/or physical entities. It can be transmitted through a physical medium or forces. It has no *objective* meaning. A particular set of meaning may be assigned by the receiver and/or sender. If there is no formal agreement between the sender who creates the information and the receiver who in turn decodes it, on the process of coding, the encoded and decoded semantics may not match.

Intelligence. Ability to deal with knowledge in *conceptual universe (q.v.)* efficiently.

In vitro. Environment outside of an organism. It is usually used for biomedical experiments of which conditions are simulations of the ones of corresponding organismal reactions.

In vivo. In a living organism; the authentic environment of a biomedical process.

Knowledge. Relations between *entities (q.v.)* and between them and their *qualities (q.v.)* and between their qualities themselves in a *conceptual universe (q.v.)*.

Knowledge module. A knowledge component, a semantic primitive, a system of semantic primitives, or entity. An *entity (q.v.)* is a conceptual result of its qualities and (more importantly) of relations and aggregations of other entities or knowledge modules. A knowledge module is a unit of knowledge representation. They are arguments in predicates in logic or nodes in Ontological Networks.

Knowledge representation. Coding knowledge into information in an established way so that some other agent may retrieve it by decoding.

Knowledge sharing. Being able to decode the information in the same way.

Medium of information transmission. It is either physical or conceptual. Medium is a platform on which knowledge is represented in a certain type of information; e.g., light is a physical medium for transmitting visual information; a natural language is a conceptual medium for transmitting the informal conceptual knowledge represented in it; a Multifaceted Ontological Network is a conceptual representation medium for transmitting the formally structured knowledge.

Metaknowledge. Knowledge about the representation of knowledge.

Metaphysics. A part of philosophy inquiring questions about physical and conceptual beings.

Mereology. A discipline of analysis, in which systems are decomposed into its parts (in contrast to be decomposed into their subclasses as in set theory).

Model. Representation of certain aspects and properties of a system.

Multifaceted Ontological Networks. Ontological Networks integrating different views. A Multifaceted Ontological Network is a conceptual knowledge representation medium *(q.v. medium of information transmission)* for transmitting formally structured knowledge *(q.v. formal knowledge, formal knowledge representation)*. It yields formal knowledge since all concepts *(q.v.)* and their subconcepts are required to be fully connected (fully defined) up to the extent of existing knowledge. The resulting picture is a "complete" conceptual universe *(q.v.)* at certain level. A Multifaceted Ontological Network may embody different perspectives of the same set of entities. For acquiring knowledge to represent and accessing the represented one, Multifaceted Ontological Network provides an interface called *facets (q.v.)*. Through each facet, an agent (a human or a software system) may access certain parts of a Multifaceted Ontological Network.

Ontological Networks. A knowledge representation platform of the methodology called Ontological Systems Analysis and Synthesis. The philosophy behind this methodology implies that formal knowledge can only be achieved by representing the "complete" conceptual universe *(q.v.)* at (up to) a certain ontological level. An Ontological Network represents knowledge in knowledge modules *(q.v.)* each of which is "completely" represented. (The completeness is

obviously a function of available knowledge). This yields *formal knowledge (q.v.)* and *formal knowledge representation (q.v.)*. The building blocks of an Ontological Network are nodes and arcs as in a semantic network for entity representation and a hierarchy of frames for representation of qualities of those entities.

Ontological Systems Analysis. The OSA is an analytical methodology which views existing systems in terms of their entities and the relations between those entities. This analysis is propagated in this thesis to the synthesis phase, OSS, in which analyzed knowledge is reorganized and represented. The output of that phase as a knowledge representation is acquired by the OSA and compared with the existing real world knowledge. The two systems (natural and artificial or the real and its representation) are analyzed in parallel and mismatches are propagated back to the OSS phase for reorganization. The analytical criteria are summarized in *Ontological Systems Analysis and Synthesis (q.v.)*.

Ontological Systems Analysis and Synthesis. OSAS is a methodology proposed in this work for representing scientific knowledge. OSAS has two integrated components OSA (Ontological Systems Analysis) and OSS (Ontological Systems Synthesis). It involves the following methods:

- Identification of different organization levels of a system.
- Analysis of the higher level organizations on top of the synthesis of the lower level ones.
- Top-down analysis of organizations at each level for extracting all essential (formal) relations between their entities, and decomposition of high level aggregations into their components in every useful abstractions.
- Identification of *all* classes of substructures and subprocesses existing in the *conceptual universe (q.v.)*.
- Establishment (the synthesis) of all essential relations between entities completely throughout the organization.
- Analysis of qualities (data) of entities. Analysis and synthesis of relations between descriptive and numerical qualities, as well as between fuzzy (numerical) data and more precise or more involved ones.
- Identification of users and their representation (degree of abstraction, aggregation, classification, and idealization) needs, and establishments of facets for fulfilling those needs.

- Refinement (resynthesis) of the representation of organization by closed feed-back looping between analytical and synthetical processes.
- Application of reasoning mechanisms to the resulting formal knowledge representation, and searching for undiscovered implicit knowledge (unidentified relations between entities) within the represented knowledge.

Ontological Systems Synthesis. OSS is a methodology for reorganizing the existing knowledge on a representation system called a Multifaceted Ontological Network. It is performed based on the output of Ontological Systems Analysis (OSA) and feeds its output, knowledge representation, back to the OSA by closing the feed-back loop.

Pragmatism. According to Charles Sanders Peirce, it "is a method of determining the meanings of hard words and abstract conceptions." (Collinson 1987)

Quality. It illustrates a property of an entity, such as weight, color, speed etc. It has a wide spectrum of precision and formats. For example, a color of an entity can be stated as "yellow," 560–610 nm or a collection of wavelengths forming a distribution curve and a spectrum.

Reasoning. Producing new knowledge using rules of logic on the represented knowledge.

RNA Ribonucleic Acid. It is a string of nucleotides that contains genetic code (like DNA).

Scientific ontology. A part of metaphysics, dealing with relations between beings. Usage of ontology in artificial intelligence is mostly in its more restricted sense such as a classification or taxonomy. In our work we use it in its original meaning.

Semantics. Meaning decoded from information.

Sensing. Being able to receive and decode a piece of information via receptors and inborn skills encoded in our genes.

Submodel. A model nested in the main model. A facet of the whole.

Syntax. Symbolic code of information.

Understanding. Being able to decode information without doubt about the correctness of the decoding process, and extracting to semantics which has some patterns similar to the knowledge already had, so that the acquired semantics can be incorporated with the existing knowledge. Understanding is not a single state but a degree of perception and conceptualization of the world represented in our mind. It is directly proportional to the degree of decoding and the depth of knowledge about the decoded context (see decoding).

Her şeyimi borçlu olduğum canım Anne ve Babacığıma...

(To my dear mother and father, to whom I owe everything)

CHAPTER 1

INTRODUCTION OF THE PROBLEM

I study artificial intelligence since it seems to me that mine (alone) is not sufficient.

Actually, the sentence above could be my entire introduction, but since this is a formal work, it is expected that I will express concepts in a more explicit manner. Even though I believe I have already explained everything, people who read this sentence, like yourself, might not agree with me. Why? If this were a piece of work in literature, neither you nor I would bother with that introduction, because the ultimate value of an artistic work is how it makes its consumers think. But this is not art. I claim that this is a scientific study and my hope is that you will agree with me when you finish reading it. In contrast with art, scientific communication is desired to be plain, simple to understand, and *unambiguous*. The duty of a scientist is not only to generate honest information but also to pass it with minimal loss to the scientific community. The scientist should be explicit in his explanation. He should be sure that the information he produced would be understood as exactly as he intends.[1]

[1]I too face the problem of sexist usage and expressing myself in a politically correct fashion; however, using at every instance *he/she* or *one* for the pronoun of the third person singular, seems to me that it either distracts the attention or looks artificial, as it is in this sentence.

Is this possible? In terms of information theory it is not possible to pass a message through a medium and have it received by someone without any loss of meaning. Despite all entrenching efforts of a scientist towards making his work explicit, some information will be lost. The success of his communication depends highly on the medium which is used to pass the information. This problem has been first introduced by as the *semantic problem* (Shannon and Weaver 1964). The essence of the problem is that there is no way to interpret a message, such as a sentence, *exactly* as the originator intends. The original meaning decays slightly even in the most unambiguous sentence.

We know very well how difficult it is to conduct biomedical research. So much effort, so much pain... After all that, it is frustrating to face the semantic problem — not to be able to pass the *exact* results of that painstaking research. The only thing for a scientist to do is to explain everything as plainly and rigorously as one's writing skill enables, and the rest is just a hope that other scientists will understand that concept in the same terms. This simple but frustrating fact shows us that the semantic problem is very important to attack because the solution would help not only biomedical scientists but everybody who has to express observations rigorously.

In this research, we think that very important clues are revealed for expressing things in such a rigorous manner. After his specification, a scientist should *objectively* know what he has been able to express, and consequently be confident about what others will understand from those expressions. By communicating in the way

we propose or by using appropriate knowledge acquisition tools which follow those principles,[2] a scientist can express all his observations in his research, and afterwards be confident about how much of his knowledge he has been able to introduce to the knowledge system. This way he will be sure that all of his knowledge he has introduced will be passed *exactly* to the receivers.

1.1. Ideal Definiteness and Flexibility in Communication

The process we desire to create resembles mathematical modeling. We know that mathematics is such a unique communication medium through which mathematically formulated domains can be passed without any loss of meaning. Skilled scientists can decode those formulae and acquire all concepts they contain. While it is theoretically possible to mathematically formulate a complex domain, such as medicine; however, in practical sense it is too hard. In addition, expecting a biomedical scientist to formulate all his observations in mathematics is not fair either, as they would be a great loss of information in this formulation itself.

By defining the ideal state of definiteness in knowledge representation in mathematical terms, we establish an aim of creating a methodology through which we can reach that level of definiteness with more flexibility in the representation. Though it may seem that flexibility and definiteness are trade-offs, we believe we can overcome

[2] A prototype of such a *K*nowledge *A*cquisition *T*ool, called *medKAT*, for biochemical domain has been built by the author.

this hindrance through the power of computer science and technology. Computers are enabling us not only to build up rigorous information structures flexibly but also to easily manipulate those structures for many different purposes such as knowledge based reasoning and simulation.

Our problem, therefore, is to represent knowledge in an information medium (other than a natural language) that transfers knowledge without any loss. Its logical consequence is to ask the question, "How can we accomplish this?" Obviously, this is an overly ambitious aim. Therefore it is only an *aim*. The *goals* of this work are to show an appropriate, clear direction and propose solutions to overcome the semantic problem. In our research, we have seen many subproblems each of which has been either recognized as an entity in current research or remained in the shadow of the giant research problems. We have integrated those topics in order to approach our main problem.

Computer science and technology have reached a point where we can solve very complex mathematical problems with very usable, quick, and affordable commercial tools that could not be imagined few decades ago. In other words, when we formulate our problems mathematically, we may get almost instantaneously desired solutions for which scientists had to spend couple of months twenty years ago. However, as was pointed out, reaching such a mathematical formula addressing all requirements, has always been too difficult, if even possible at all through conventional methods. Since we cannot conceptualize the world mathematically *per se*, our approach is to

analyze those intrinsically complex problems in terms of conceptual relations at the start. Thereafter, we associate those relations and some of their qualities with our knowledge in mathematics, and come up *heuristically* with the resulting formula. The problem is then how to bridge qualitative and vague concepts in our mind with the quantitative, rigorous formulae heuristically. When we study mathematical problems, we somehow come up with mathematical models, but we do not know a recipe *how to do* it.

1.2. The Many Faces of an Entity

In mathematical models, there are many implicit assumptions which cannot be stated within formulae and are left to be explained in a natural language, such as English, which is intrinsically vague in terms of carrying the meaning. Today, there is no formalism to express what a mathematical primitive stands for, other than explaining it in a natural language. On the other hand, due to their nature, mathematical models are highly specific and narrow in relation to the domain. We see many variants of mathematical models addressing the same problem with different conceptualizations of the domain. Since the assumptions and simplifications are different, they are not compatible with each other, they cannot be unified into a single formula easily.

This is also true in daily life, as the same issue can be treated differently by different individuals. From an author's point of view, a book may mean a composition of work with a title, several chapters and a bibliography, whereas a bookbinder may con-

ceptualize the same artifact as a cover, pages and binding material (see Figure 1.1.). The same is true for professionals in medicine. The conceptualization of a medical

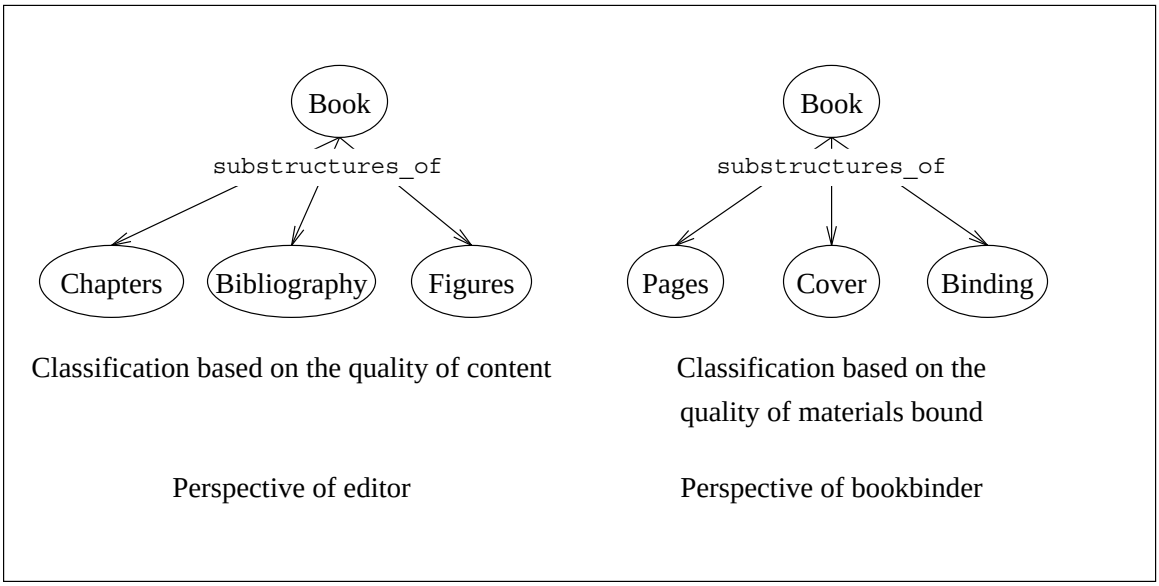


Figure 1.1. Different decompositions of an entity based on seeing their components in certain classes where each of them differ in qualities.

approach, such as examining a patient or treating a disease, may vary between an internist and a surgeon. Though the objects are the same, the conceptualizations are different. Each conceptualization reflects the truth from some perspective, emphasizing some parts more than the others, and possibly neglecting some components entirely. Such classifications and categorizations are not natural but are creations of the human mind. They allow us to conceptualize things easily and approach our specific problems more conveniently.

1.3. Many Problems under a Single Title: Knowledge Sharing

While areas of interest are becoming more specialized, the number of these specializations is increasing in terms of different classifications, each with their own specific representations. A dilemma of increasing magnitude is to solve interdisciplinary problems in which these representations are not compatible with each other. Databases and expert systems are trapped within this dilemma(Buchanan and Smith 1989; Lenat 1989), as they are unable to transfer or share knowledge with their other counterparts because each of them is designed with particular representation commitments. For example, time might be represented in a certain format, such as continuous or discrete, interval or relative, and systems with different representation formats may not exchange information.

These commitments in knowledge representation are also called Ontological Commitments since in conventional knowledge bases a system is represented (modeled) in terms of a set of ontological relations by neglecting the other sets. As will be seen in the rest of this thesis, we do not neglect the other ontological relations but combine them on the same representation platform. Our goal is to build up knowledge which can be utilized by people who classify it differently, do not share same terminology, nor share the same methodological perspective. Even though the approaches of those people may conflict with each other, the knowledge should still be in position to be shared.

The main problem is passing knowledge without loss of semantic through our information processing devices, computers. In order to facilitate knowledge sharing among different representations, our concepts must be first able to interpret the meanings of those representations. Our goal is to find out how to provide *consciousness* for our computers so that they can understand what they are doing. For example, a computer ought to *know* what a "table" is in terms of its structure and possible functions, rather than capture it as a sequence of five ASCII characters. The latter is the state of the art performed by some hypertext and database systems, which do not help much in terms of compensating our inefficiency in processing semantics hidden in an exponentially growing information bulk.

We should (somehow) let the computer *know* what particular information means. There are many subproblems in terms of incompatibility:

- The same entity can be classified differently by different professionals.
- Different professionals may have different priorities and prospects with respect to the same entity.
- Different professionals may require different levels of approximation or resolution in the representation of the entity.
- Different professionals may use different terminology to express similar concepts.

These problems form the superset of incompatibility problems between knowledge

bases that differ in ontological design commitments. These incompatibility problems must be overcome so that everyone can conveniently access and utilize knowledge of the current state of science (Musen 1992). This may be accomplished through a new knowledge representation methodology. This thesis presents studies intended to form parts of such a methodology.

CHAPTER 2

FROM INFORMATION THEORY TO PROBLEMS OF KNOWLEDGE

The terms *data, information* and *knowledge* are not well defined. In daily life, we use them in interchangeably. Computers, however, need unique information for every piece of meaning, so that they can establish a mapping between them. Therefore we, as computer scientists, need to define these concepts and use them consistently throughout the thesis.

2.1. Information

We define a piece of information as a sequence of symbols or signals of any kind. It can be carried via a physical medium such as ink on paper, electromagnetic waves in various ranges such as visual, audible, radio, and so on. If the transmitter is an intelligent agent, such as a human, information may be sent not only via a physical medium such as a voice but also via a conceptual medium such as natural language, or as a statement in logic or as a mathematical formula. The piece of information may be formed by a transmitter altering the physical medium in a certain manner to send a certain sequence of primitives provided by a conceptual medium. This is called *encoding*, representing knowledge in a piece of information. The receiver should

be capable of sensing the piece of information, and be able to decode it in order to understand and acquire knowledge from it. (see Figure 2.1.).

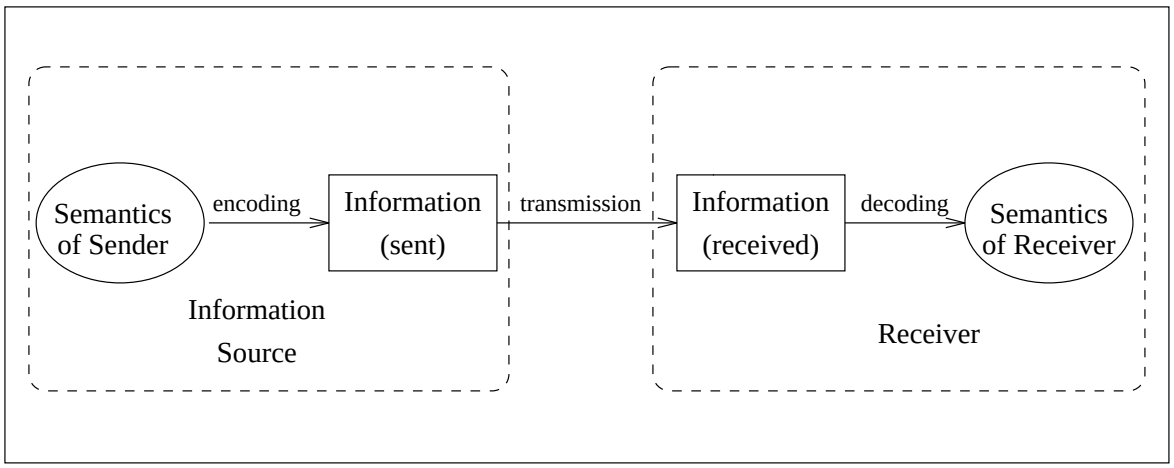


Figure 2.1. Basic elements of information transmission or communication.

What are examples for information? A sentence as well as a word are information. Blushing, becoming red in face under acute stress, is also information. Certain orders of bases on a ribonucleic acid (RNA) string in a cell, or electromagnetic waves produced by an inducer are other examples of information.

2.2. Semantics and the Semantic Problem

Information itself has neither a context nor a meaning. However, the generator called the sender or transmitter may associate that information with a certain meaning, which is called *semantics*. Similarly, the receiver can associate that information

with a certain semantic meaning as well. However, both are subjective associations as the same information may generate different thoughts and feelings in different people or beings. In order to be able to decode information we need to be familiar with the type of information being sent. For example, information exchange via body languages between birds rarely makes any sense to a person illiterate in zoology (see Figure 2.2.).

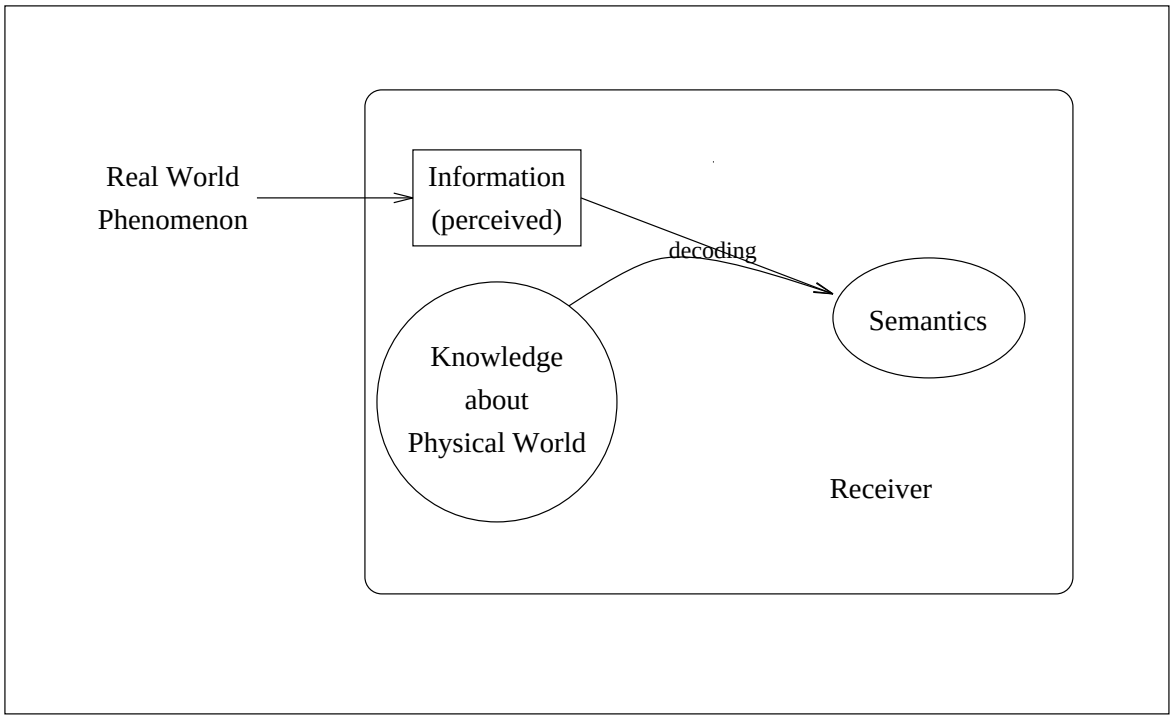


Figure 2.2. Observing physical world and decoding perceived information.

This semantic problem also may be expressed in terms of physiology. For instance, in a stress situation, adrenal glands in our body release hormones such as

epinephrine into the circulating blood so that it is distributed to all organs. However, every organ acts differently according to its functions and to the quantity of relevant adrenergic receptors (for epinephrine) it contains. In other words, those organs get the same information, but that information is sensed and interpreted differently.

Of course, the semantic problem is most acute in natural language. Although information itself does not contain meaning, the user interprets it with his background knowledge and assigns meaning to it. Because of the informality of this interpretation, semantics is dependent on various things, particularly on the context. The same sentence in a natural language may mean totally different in different context. For example, the word "blue" carries different semantics in different contexts; it may point out the color of blue as well as a quality of being depressed. This richness of variety in semantics jumps in quantity when we use such idioms and metaphors in our communications. This is the result of overloading the information modules, such as words or phrases that we use in our *Semantics Exchange* or *Verbal Communication.*

We can verbally communicate with each other via some standards about relations between information and semantics which are established throughout the history of mankind and taught to us during our childhood and school days. While this mutual understanding works on simple matters, it unfortunately does not always work with complex and abstract concepts, which might be understood differently by various people. This is the origin of confusion and misunderstanding. The sender intends

certain semantics by sending particular information, but that information may be interpreted differently by the receiver.

This relativity of interpretations is closely bound to the medium through which information is transmitted, and to a certain extent it is also bound to the style — i.e., the way the medium is treated. For example, in order to prevent misunderstanding in communication, the sender may attempt to send the information redundantly so that a transmission error can be caught by comparing this redundancy. We also try to use this principle by expressing the concepts in this work in various fashions and with many examples. However, this treatment of communication medium is limited to the definiteness of the medium itself. In our case, the medium is a natural language expressed in a written publication. Tools for transmitting information via natural language (e.g., ink and paper) are sufficient for preserving the originality of the work but the natural language itself is not sufficient to express the intention and to transfer it to the receivers by keeping its originality.

2.3. The Three Levels of Communication Problems

In order to investigate this issue further, we should consider the perspective of information (or communication) theory. The communication problems are categorized by Weaver into three levels (Shannon and Weaver 1964): The first level, called the *technical problem*, concerns the accuracy of the signal (or symbol) transmission. This is a technological problem in communication and has no significance in human-

computer interaction. The second level, called the *semantic problem*, concerns the relationship between of meaning intended by the sender within one's information and that interpreted by the receiver from that information. *This* is the problem we intend to attack. The third level, called the *effectiveness problem*, concerns the effectiveness of the received meaning in terms of conducting the receiver in the desired way. This can be illustrated with an advertisement on TV, of which effectiveness is defined as selling certain item and imposing a certain way of acting (or living) that is profitable to a particular company.

The effectiveness problem overlaps highly with the semantic problem. In computer-human interactions, the semantic problem and the effectiveness problem are almost identical. If we know exactly what to do, and if we don't make any errors, computers respond according to our wishes. However, in large software systems or highly complex algorithmic designs, the distinction between effectiveness and semantic problems may be clearer. Although the procedures written are correct, they may not necessarily be efficient or reliable enough. On the other hand, in the human1-computer-human2 interaction, where the computer is a bridge between people, the effectiveness problem involves conducting people through the knowledge represented in computers. This is the concern of the professionals of Computer Assisted Instruction, an area of AI and cognitive science. Nevertheless, the unconscious part of this conduct (manipulation) is out of our context. Its only relevant part for us may be to foresee new opportunities

which might appear when the underlying scientific systems would be treated in the way we propose.

2.4. Data: A Specific Type of Semantics

After treating the concept of information in its information theoretical sense, we now need to define a specific type of semantics, *data*. Data is a measure of a property of an entity, such as a thing or a process. It may be in a predefined format, such as in meters, inches, grams, or ounces, if it is quantitative. If it is qualitative however, then it should still be translatable to a certain quantity, and therefore to a certain format. For example if we measure length in meters, then "X is long" indirectly indicates a fuzzy range of quantity in meters. For another example, "yellow" is a color, a quality of matter in reflection of light within visible range. For "yellow," the wave length of reflected light is around 560-600nm. In other words, "yellow" implicitly indicates a measure in nanometers.

Qualitative descriptions usually have more implicit assumptions than the ones the quantitative descriptions have. For example, if two pieces of information about the same table are "The table is high", and "The table is 80cm high," then the former one might assume that an average table is 60cm high and if the height of the table is between 45 and 75cm, it is normal; whereas, the latter has no such an implicit assumption. For the yellow example on the other hand, there is no difference between two pieces of information, "yellow" and the reflected light in the range of 560-600nm,

as both probably indicate sun light is the source of light and other than that specific range, all visible wave lengths are absorbed. So, both representations have the same assumptions. The qualitative representation formats are usually less precise than the quantitative ones, but again not necessarily always. For the table example, this is true, but not for the yellow example. The term "yellow" might sometimes be more precise than some numeric format. Consider this information: "This wall reflects the wavelengths around 1μm." 1μm is equal to 1000nm and considered in range of infrared wavelengths, which is out of the visible scope of electromagnetic spectrum. This information is actually much more approximate than the description "yellow."

In short, there is an implicit connection between every quality and quantity, which needs to be made explicit within the computer. Without accomplishing that, we cannot fully pass the intended semantics to the receivers at the other side of the communication line when qualitative descriptions are used.

2.5. Knowledge

"How do we know?" is the classical question in philosophy. Though there is no agreement we shall attempt to define it within our information-theoretical context. If the information source is reliable and if we have no doubt about our information decoding process, then the semantics of that decoded information we acquired is knowledge.

2.5.1. Knowledge seeks reliable information

Let us analyze this definition within a scenario. Imagine you are in a room for many days, but you are not aware of how many. The room has no window, and you have no connection to the outside. In these circumstances you have found a watch and would like to at least know what time it is. The watch shows 1 o'clock, but after careful observation on the watch you have discovered that watch does not run well. Now 1 o'clock, the information you received, has no truth value since your source cannot be considered reliable anymore; therefore you cannot *know* whether it is 1 o'clock or not. It is not knowledge for you (Lehrer 1990).

2.5.2. Knowledge requires understanding

The second condition in the definition of knowledge is the ability of the receiver to decode the information. Let us analyze it in another example. Consider a microbiologist in a laboratory observing some biological phenomena under microscope. Let's assume that through all his observations, he cannot see any pattern he is familiar with in order to describe those observations (see Figure 2.3.). In other words, he observes something, gets some information but cannot decode that information in a way that makes sense to him. What he would feel at this point can only be described as "puzzling." He does not understand what goes on under that microscope, so therefore the information he received is not knowledge. The degree of the ability to decode information determines the degree of understanding the semantics of that

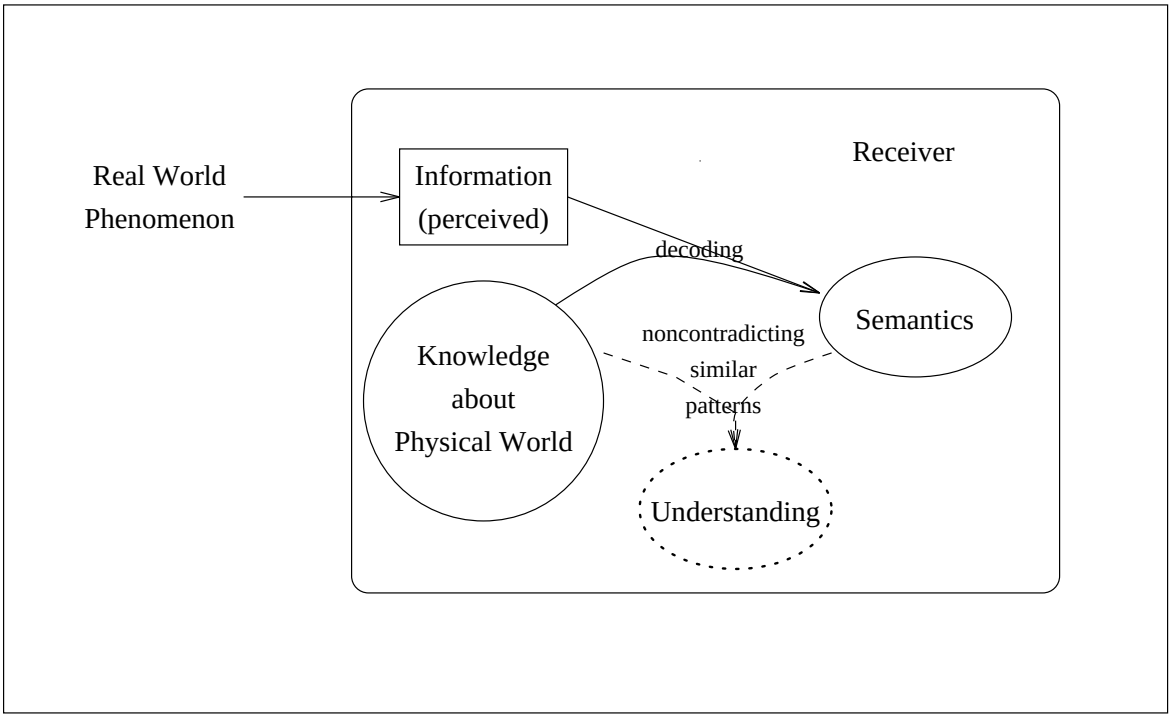


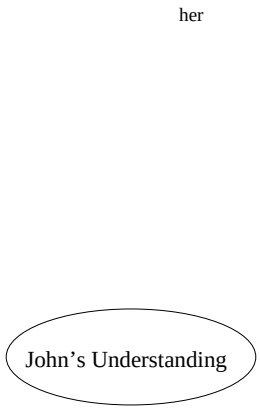
Figure 2.3. Understanding observations.

information. In this case understanding is a judgment about the ability of oneself; however, it also can be about some other person's ability.

2.5.3. Understanding as an effectiveness problem

Making a judgment about some other person's understanding is a more intricate problem. Let's analyze this within another scenario. Imagine a literature class in a high-school where students read and try to understand poetry. Assume one of the

students, John, is asked to interpret a single line of a poetry by Mrs. Buchanan (see Figure 2.4.). After he explains his interpretation, she might judge that his interpre-



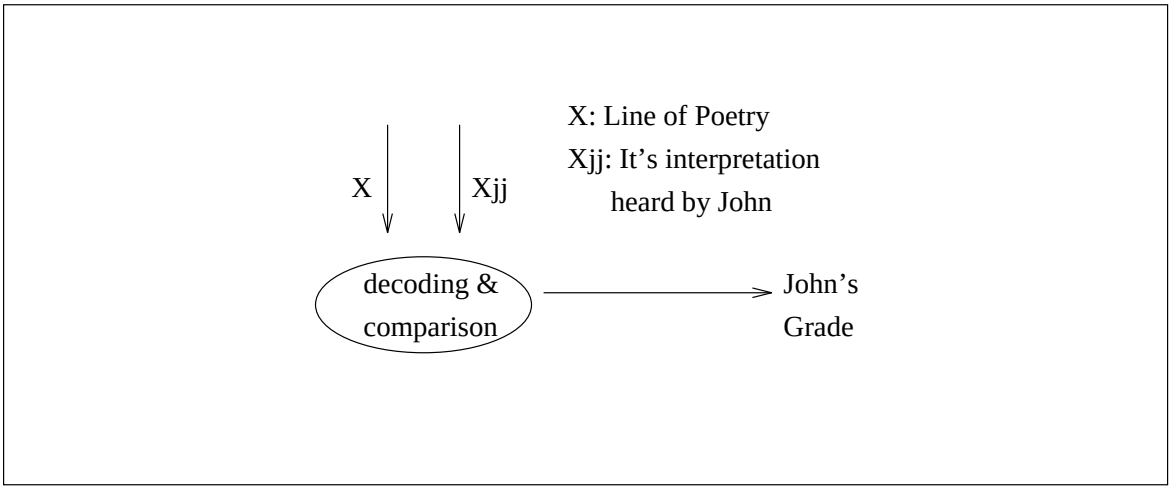


Figure 2.5. Mrs. Buchanan's point of view.

she knows how good *his understanding* of that poetry line is. From John's point of view the grade he received via his interpretation is his *effectiveness* on Mrs. Buchanan (see Figure 2.6.).

As we have seen in those examples, we need to have a reliable information source and we need to be able to understand the semantics thoroughly in order to acquire knowledge out of the information we received. Thorough understanding in this context corresponds to being free from doubts about information decoding, and seeing a tight connection between our existing knowledge and the semantics extracted from information by decoding. However, thorough understanding here does not apply to the implicit (hidden) semantics but only those that are explicit. Therefore, it does not mean to discover the nature of the context *thoroughly*. This definition of

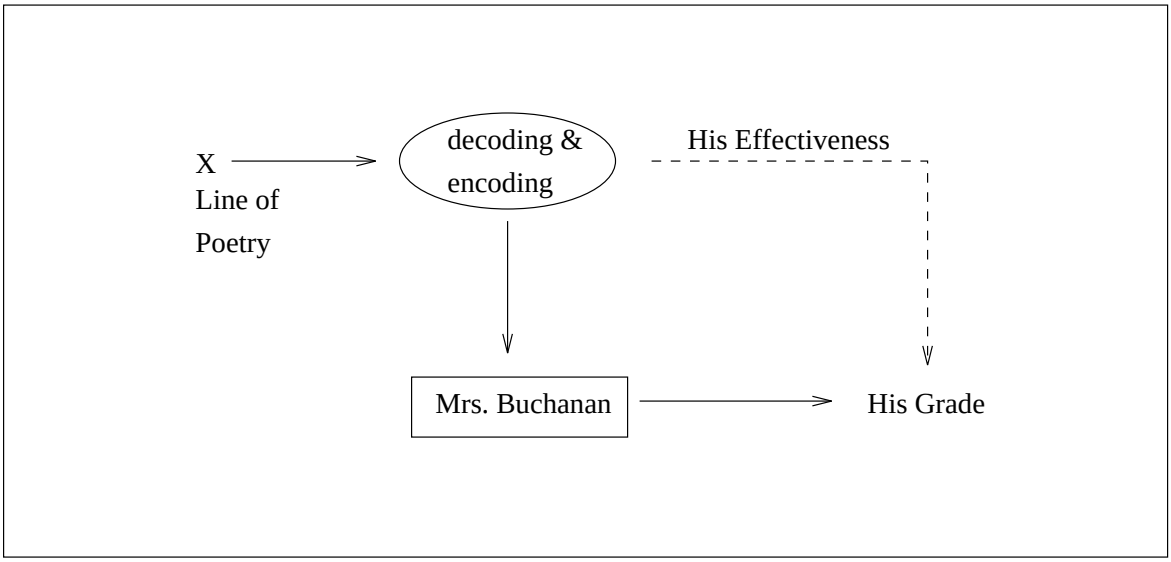


Figure 2.6. John's point of view.

knowledge is not decisive (Dretske 1981), and restricted to our conscious knowledge;[3] however, it should suffice for scientific knowledge which scientists generally attempt to communicate as explicitly as possible.

2.6. Understanding in Computers

If we analyze the John & Mrs. Buchanan example, we see a complex communication between two people. There are many instances where the semantics potentially

[3]From unconscious knowledge, we understand the knowledge encoded onto our genes controlling our cellular and organismal activities at biochemical level with neither our will nor our awareness. In his lectures, (Popper 1990), Popper treats knowledge at large which incorporates both conscious and unconscious knowledge that I distinguished with pragmatic reasons by considering scientific knowledge is a subset of our conscious knowledge.

may alter, which are the one encoding process and the three decoding processes (see Figure 2.4.) As we pointed out previously, the problem arises because of overloading of the information modules we use. From this perspective, we can see that if we are able to encode any concept in a unique manner and standardize this coding protocol, the code could easily be decoded into its original semantic (see Figure 2.7.). This hap-

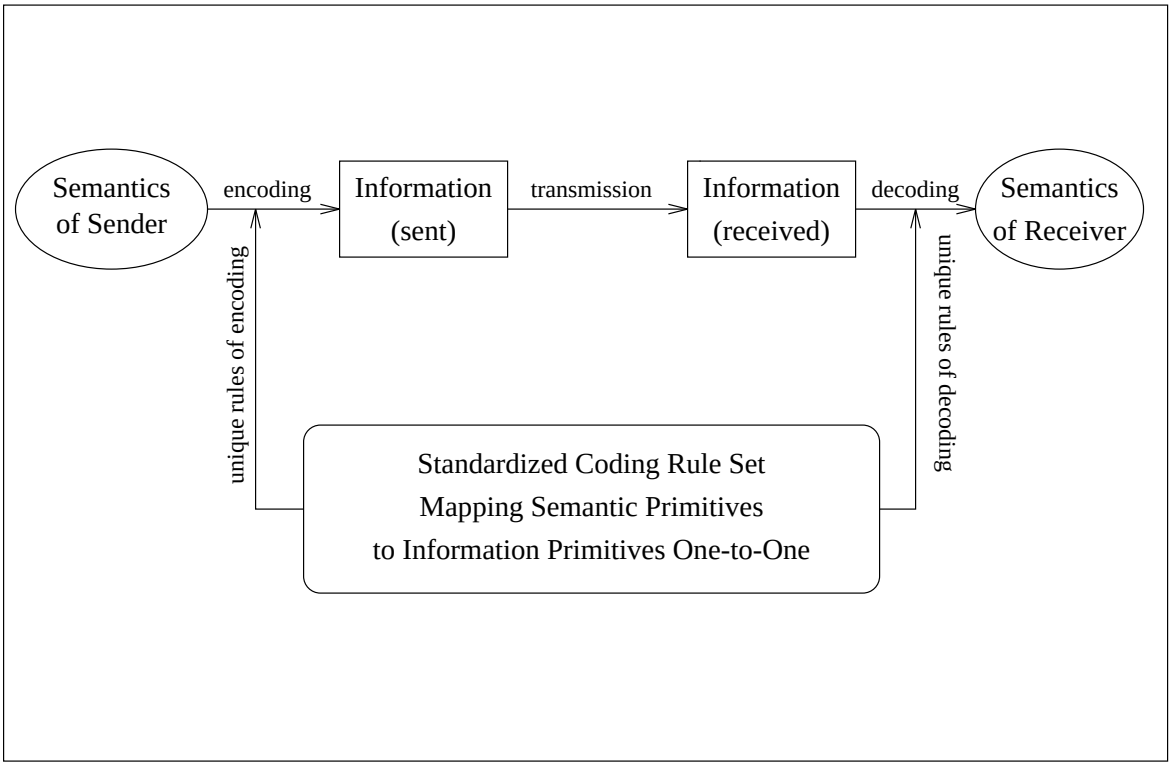


Figure 2.7. Preservation of semantics by one-to-one mapping between semantics and information primitives in coding.

pens in mathematical communications after agreeing upon the meaning ofparameters

and variables in formulae transmitted. The same type of communication between a person and a computer, however, occurs differently.

Our perspective with respect to computers are very similar to the point of view of John in the example. In this example John does not concern himself with the mental processes of Mrs. Buchanan, see Figure 2.6. All his actions are toward achieving a good grade, which may be seen as his effectiveness in terms of Information Theory. We mostly share this view when we deal with computers. We send it some information and expect something from it. Our expectations and the output we get determine our effectiveness when using the computer (see Figure 2.8.). The difference

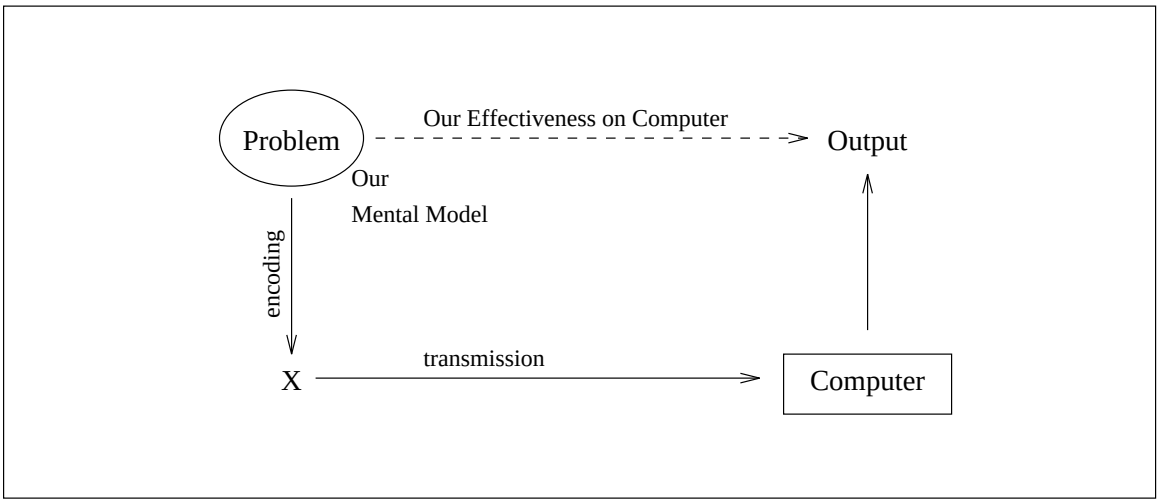


Figure 2.8. Our general view in computer communications.

is that in this case, the computer is under our control; we know exactly how computers operate, and with this knowledge we can get the desired output if we transmit proper

information as input to our computer. The problem is its input requirements and our expressions may differ so much that the precise encoding might be too hard for us, especially in complex cases. This is similar to communicating with a person who understands neither our language nor any sign language we know; that would put us in an helpless position. In other words, the problem is not with the computer side but with our coding process. We are not able to talk in the languages of computers effectively. Even when computers were solely conceptualized for computation, this difficulty was seen and higher level languages have been progressively built. These were attempts to make computers communicate in ways closer to our own languages. For primitive concepts such as iteration, we have accomplished that already. However, we still cannot make computers give us specifications of things such as chair, table, automobile, etc., at desired levels of precision, granularity, scope, and combination, since all those levels are independent variables which change according to our varying needs. We cannot capture them in a synthesized manner on computers, although we can analyze specific parts of them one at a time and perform very complex tasks on that specific part. In other words we are still not able to express our world in a *synthesized* manner.

2.6.1. The natural language approach

One way the researchers have chosen to reach this goal is to make computers understand our complex natural language. This is highly difficult, since natural lan-

guage is itself intrinsically ambiguous. We ourselves frequently interpret the same information given to us in English differently. In addition, abstract concepts such as knowledge, information, etc. are not well understood; i.e., those words are almost always conceptualized differently by different people. Though we don't think natural language understanding is impossible, it is too hard to solve intricate problems, such as philosophical inquiries, or political problems, unless we restrict ourselves to a *formal* subset of our natural language.

2.6.2. The ontological approach

Another way to utilize computers with a high rate of effectiveness rate may be the one we propose here, that is, let the computer understand all of our concepts. Almost every concept is an aggregation of many variant subconcepts and factors. The computer should always be able to single out a unique set of semantics after a certain amount of analysis. In order to accomplish that, we have to provide it the whole arborization of subconcepts and variants for every concept important to us. This work is enormously complex to carry out with paper and pencil. However, with the help of a computer's thoroughness, we can describe those concepts along with their subconcepts in a recursive manner, and incrementally build our ontology for every important discipline vital to us. Clearly, in order to overcome this complex problem we need to have a solid methodology. We call that methodology *Ontological Systems Analysis and Synthesis,* and have started to research its essential properties.

2.7. Conclusions of Our Observations

When we analyze genetic knowledge, we see that Nature has already overcome *technical, semantic* and *effectiveness problems* of communication. According to our observations in genetic code, representing knowledge in an information package (i.e., encoding) and acquiring knowledge out of that package (i.e., decoding) is performed in a formal and rigorous manner; i.e., the semantics is indubitable and the process is universally unique. Knowledge in organism is represented by conforming to a certain formalism[4] and is conserved in all living organisms.[5] If we consider that the knowledge in our genetic code is many orders of magnitude greater than our current scientific knowledge, then we could, at least, agree on the conclusion that the best way of conserving and utilizing knowledge, so far we know, necessitates a formalism for its representation into certain information primitives as well as for information decoding and the execution of the semantic. This is probably the only pragmatic way for us to overcome the knowledge problem of our era. Through an efficient formalism of knowledge representation, we may be able to stop the uncontrolled growth of information. For this purpose, we need to organize our *whole* scientific knowledge into that formalism so that *all* new knowledge can be associated with pieces of represented knowledge for higher utilization and reuse.

---

[4]The representation primitives are triplets out of four different bases and specify 20 aminoacids, where the combination of those aminoacids as a polypeptide chain along with other organic molecules build the basic molecular structures of *all* living organisms.

[5]Every base triplet specifies a very same aminoacid throughout all species.

In this analysis, we have seen that we as human beings are an exemplar source of *implicit* information to ourselves; knowledge is hidden from ourselves within ourselves. Our main occupation in natural sciences is trying to make it more *explicit* — putting our unconscious knowledge into our consciousness. These observations show us that we need to formalize our complex concepts in our mind and scientific knowledge. In order to do that, we have to make every single property of those concepts as well as every slight difference between them explicit. In our knowledge representation system, every relation (information primitive) *has to* correspond to a unique semantics as a primitive of represented knowledge. Using only those semantic primitives, we have to reshape our conceptual mind in the computers.

This is a bottom-up, synthetic process. The obvious requirement is to build the bottom level, i.e., to make the semantic primitives explicit. This can only be accomplished by thoroughly analyzing the concepts of scientific knowledge at that level. This analysis is a (top-down) decomposition process. It may also involve knowledge representation from a certain viewpoint (such as classifications based on certain qualities), but it must represent all viewpoints and considerations within the same representation system. This is the only way, as far as we see, to acquire all semantic qualities of the underlying system. Only with that degree of complete knowledge we can reach useful synthesis. In this process, we should not neglect any aspect of higher aggregations; i.e., we should not limit ourselves with a certain ontological commitment or a certain viewpoint.

The rest of this thesis will be concerned with outlining a methodology for formal knowledge representation called *The Ontological Network*. As its name indicates, we are going to use ontology to analyze actual systems and to synthesize a knowledge system for representing those actual systems in computers. Therefore, we call this methodology *Ontological Systems Analysis and Synthesis*, or in short OSAS. In the next chapter, you will find the details of the methodology we have formed so far.

CHAPTER 3

A METHODOLOGY: ONTOLOGICAL SYSTEMS ANALYSIS AND SYNTHESIS

3.1. Introduction

There are many definitions of *ontology*. Probably, the most general one is *"Ontology is theory what exist."* (Urmson and Ree 1989). We need ontology in order to formalize our knowledge, by establishing rigorous relations between its components and properties. This would yield unambiguous definitions for concepts that we find hard to define. If the parts of the concepts can be thoroughly and consistently defined, processing those concepts on computers could be possible. Our aim is to reach such a formalization. So far as we have seen, the best definition of ontology as we understand it is made by Günther Jackoby in his work *Allgemeine Ontologie der Wirklichkeit* in 1925 (Burkhardt and Smith 1991):

Ontology is the theory of the most general formal relations of reality.

Our aim can also be identified in terms of *pragmatism*. It is defined by Charles Sanders Peirce as follows:

Pragmatism is a method of determining the meanings of hard words and abstract conceptions.[6] (Collinson 1987)

3.1.1. The need for rigor in scientific descriptions

In natural sciences, knowledge is described in an *ad hoc* manner. It is a common belief that precision or rigor on the one hand and expressiveness on the other are tradeoffs. Seeking its expressive power, scientists always document their observations in natural sciences in a natural language such as English. Since those scientific documents lack necessary rigor, we cannot process them in computers.

This is a major disadvantage, as software technology today makes overwhelmingly tedious and intricate things easier to grasp. This fact can be appreciated if it is considered that computers and system software close the gap between binary code and English-like programming language representations. We need to build additional system software for closing the "last" gap between programming languages and our intricate concepts (see Figure 3.1.). Although it is not an easy task to accomplish, no one would probably say that it is impossible, especially after observing the previous success of computers in this matter. Therefore, the "expressiveness-rigor tradeoff" is not valid any more given the state of computer and software technologies today.

[6]From this perspective probably, Peirce also developed his *existential graphs* as a logical notation similar to today's Semantic Networks (Sowa 1991).

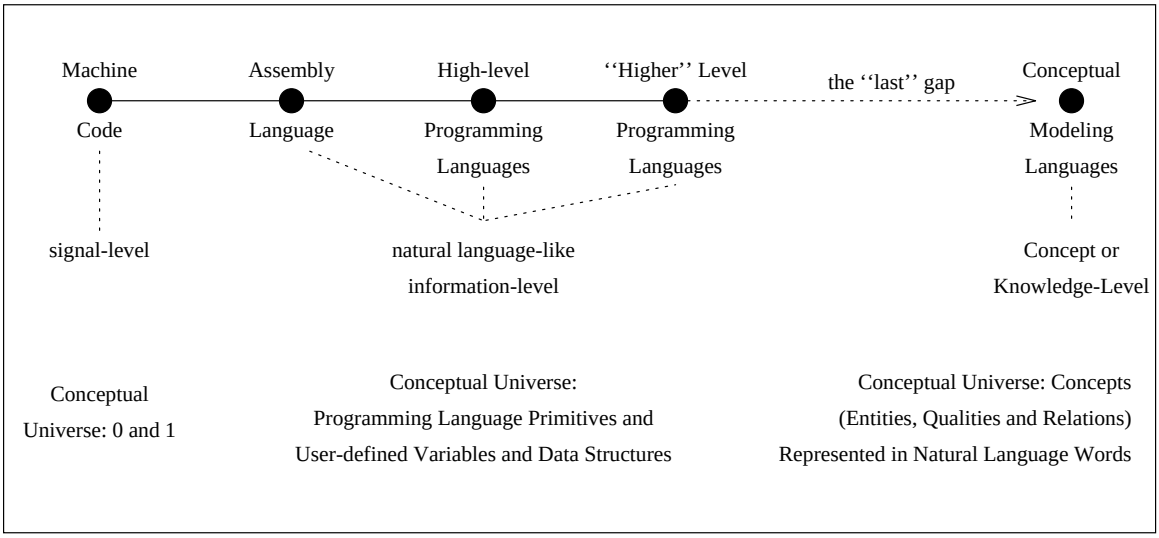


Figure 3.1. Conceptualization levels of computers.

How can computers understand our concepts[7] in natural sciences? As we analyze each concept deeper, we recognize its vagueness.

> *The question of interpretation has been unduly neglected. So long as we remain in the region of mathematical formulas, everything appears precise, but when we seek to interpret them it turns out that the precision is partly illusory. Until this matter has been cleared up, we cannot tell with any exactitude what any given science is asserting.* (Russell 1965)

[7]There are two main approaches in artificial intelligence, symbolic representations and connectionist representations. In this work, the former is considered.

3.1.2. Variation in conceptual models

The vagueness of concepts and variances in their interpretations are usually because of the different assumptions in their conceptualization, synthesis, or construction. Synthesis in this context means to build a single conceptual entity by *aggregating* some other conceptual (sub)entities. "Entity" implies a semantic primitive or knowledge module. It can be a structure or a process.[8]

This is a problem with many obstacles. When we decompose an entity into its components, we may see that these building blocks are not identical to those in different interpretations (i.e., in *different decompositions* with different conceptualizations). The same idiosyncrasy may be observed in relationships between an entity and its components; different interpretations may imply *different relationships* between entities and their components. In addition, different views may place the same entity in *different classes*.

If we can identify these different assumptions or commitments we may overcome the problem through a methodology based on an old tradition in philosophy, called ontology. The methodology in this work is therefore called *Ontological Systems Analysis and Synthesis* (OSAS).

Concepts as single words have no meanings in isolation. We may conceptualize them in their definitions in which they do not take part. Definitions are aggregations

[8] "Entity" throughout this work does not refer to qualities, i.e., data.

of some concepts and relations between that aggregate and some other concepts. The latter part determines properties and qualities of the defined entity. We can understand concepts only in terms of other concepts and relations. For example, take this sentence: "The entity X is 1.2 meter long." We can understand the length of X only in terms of other concept which in this case the reference unit meter. We can conceptualize the length of X if we can recall how long a meter is. Then we can deduce it is 1.2 times one meter long. Take another example, "X is computer." If we try to understand what kind of *thing* it is we need to consider other concepts such as keyboard, monitor, unit's box, ALU, memory, disk for secondary storage, system software, etc. The computer is nothing but the aggregation of those components. If two people agree on those aggregated components then they would agree what a computer is; otherwise, one might think of a personal computer while the others might think of a mainframe, a calculator, or a home entertainment tool.

3.1.3. A unified conceptual universe based on natural sciences

We need to use the same method to make our computers *understand* our concepts and eventually our *conceptual universe*. If we symbolize our concepts in unique words in computers, and can relate them completely and consistently, then we can say that our computers understand our concepts.

It is so simple and it is so hard. Simple because we know what to do. Hard because we know that to provide consistent relations for a complete set of concepts in

our conceptual universe has not been done before. All efforts in philosophy show that people do not agree on the definitions of concepts. This is partly because they do not perform this activity in systems as rigorous as our computers, and partly because they differ in linguistics and traditions of usage, which we do not consider here. Another reason for their failure is they deal with their subject, humanity, at very high levels such as politics, ethics, linguistics, cultural artifacts and psychology, which we do not consider at all. Given the difficulties that philosophers have faced throughout history in this area, we know now that we should avoid speculation and use materials for which there is little doubt and common agreement. We need to start representing our world within our computers starting from the very fundamentals of natural sciences, since they include more *objective knowledge* than any others. As consequences of this objectivity, are more clearly stated, more elaborated, more tested, and therefore, they are more certain.

> *Objectivity is not the result of disinterested and unprejudiced observation. Objectivity, and also unbiased observation, are the result of criticism, including the criticism of observational reports. For we cannot avoid or suppress our theories, or prevent them from influencing our observations; yet we can try to recognize them as hypotheses, and to formulate them explicitly, so that they may be criticized.* (Popper 1983)

The basics of natural sciences are the most studied subjects and therefore they are the most open ones to criticism. They are based on unbiased observations. The biased ones are consecutively eliminated by the others through refutation. The fundamentals of natural sciences are continuously challenged by the newly built concepts

that are erected on top of those fundamentals and ostensively prove the validity of their fundamentals in every test. The resulting scientific structure, i.e., new concept on top of their basics, must be coherent with every observation and outcome, which are acquired more precisely every day. Entities of natural sciences processed through OSAS and represented in an Ontological Network, are *factually objective*, since they are based on "factual reference" (Bunge 1983).

Popper's objectivity definition does shed light onto the higher degree of objectivity of the basic natural sciences compared to the others, but moreover it also implies why we have to accomplish OSAS. We have to do that in order to "*formulate*" all of our concepts "*explicitly, so that they may be criticized*" extensively by everybody using the power of our era, the computer. Our concepts expressed in natural language texts are user-dependent and therefore vague and hard to analyze, and are relatively protected against refutation by objective knowledge. Whenever we redefine them in terms of *explicit* relations of other formally defined concepts within a Multifaceted Ontological Network through OSAS, we then let them become such objects that can be *open* to "inter-subjective or objective criticism" (Popper 1982).

The discovery of this ontological process was a turn for human being and let him structure his culture. The discovery was in two steps, first the "discovery" of language, and then that of writing. Through language, we have been able to formalize our amorphous ideas and make them objects to discuss and to improve in turn.

> [It] *makes a great difference whether we merely* think *some thought or whether we* formulate *it in a language (or still better, write it down, or get it printed). As long as we merely think the thought it cannot be objectively criticized. It is part of ourselves. To be criticizable it must be formulated in human language, and become an object.* (Popper 1982)

Our approach is an attempt to make a next step in the order of *explicit formulation* of our thoughts and concepts. It can be only objectively handled if we establish the relations we intend to use. Only in this way can we share the *exact* context. Through OSAS, we extract essential relations and use them as extensively as we can. Indeed, we do not need fancy formulations or unnecessary synonyms for describing our objective knowledge regarding our universe. Every term represents a unique concept and every concept (or entity) is represented in a unique term.

> *... every empirical science, however abstract, must contain in any minimum vocabulary words descriptive of our experiences. Even the most mathematical terms, such as "energy," must, when the chain of definitions is completed until we reach terms of which there is only an ostensive definition, be found to depend for their meaning upon terms directly descriptive of experiences, or even, in what may be called the "geographical" sciences, giving names to particular experiences. This conclusion, if valid, is important, and affords great assistance in the works of interpreting scientific theories.* (Russell 1965)

3.2. Integration of Ontological Analysis and Synthesis

OSAS has two parts: The analysis and the synthesis. They are not separate phases, though. They are performed in parallel as part of a closed feedback loop. The analysis part acquires knowledge from a scientific source while the synthesis part

processes and represents it in the Ontological Network. The result is propagated as feedback to the analysis part. If it indeed represents the actual knowledge then represented knowledge is analyzed based on necessary criteria such as *groundedness* or *consistency* as directed by the Ontological Network. If, however, it does not represent the actual knowledge in reality, then the reason is determined and analyzed in order to form a new synthesis (see Figure 3.2.). The processes illustrated in Figure 3.2.

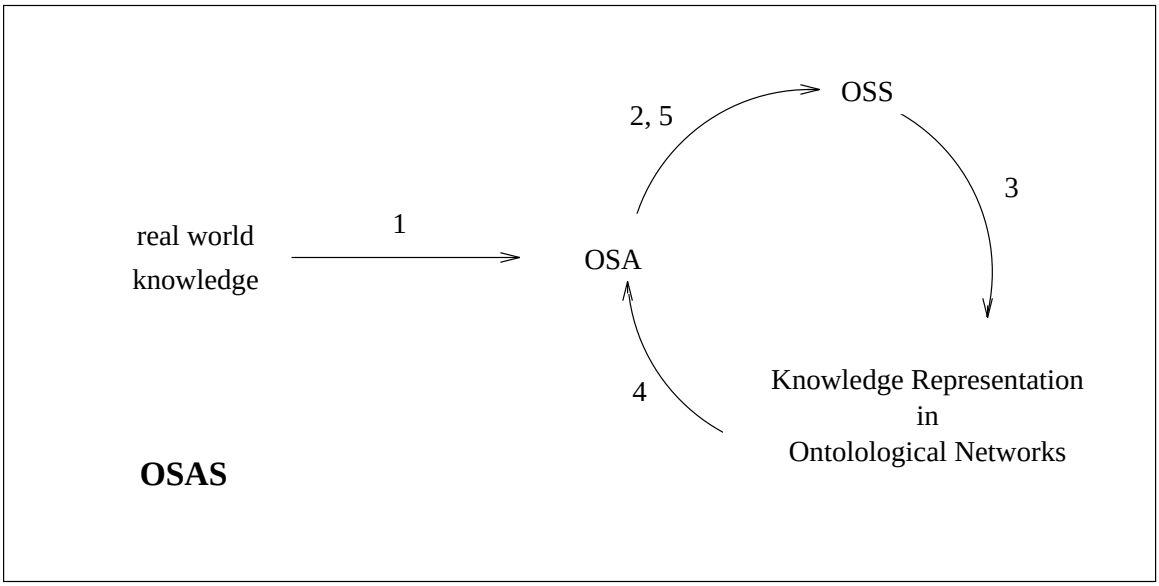


Figure 3.2. Processes in Ontological System Analysis and Synthesis.

can be described as follows: 1. Information transmission: The professional analyzes acquired knowledge ontologically. 2. The analysis are synthesized in terms ontological relations and entities. 3. The result of ontological analysis and synthesis is represented in an Ontological Network. 4. The resulting Ontological Network is analyzed by the

professional. 5. The analyses of the new state of Ontological Network compared with the new analyses of real world knowledge are propagated for a new synthesis.

This two part process is not only a philosophically "correct" approach but also an answer to the *holistic-reductionist* quarrel. Analysis and Synthesis are the methodological processes as stated above as well as representational processes in this work. Unlike many other approaches, The Multifaceted Ontological Networks represents knowledge thoroughly in various granularities. Therefore, systems are represented in both aggregates (wholes) and components (parts). Analytically acquired qualities of the components do not completely describe qualities of the whole. In addition to those distinct, partial qualities, there will exist some additional qualities to complete the description of the whole. The latter ones are related to the entity at that higher aggregation level.

The inheritance of qualities are between entities and their instances, but not between entities and their parts. For example, the entity "human" has some qualities. Those are inherited to any of its instances such as child, American, patient etc. (that is, regardless of the classification quality, which in this case are age, citizenship and medical condition respectively). However, those qualities cannot be inherited to the parts of a human being such as eyes, hair and so on. Although we analyze entities in their parts, we also represent those entities in a synthesized format. Moreover, this representation enables us to synthesize new qualities for entities based on the observations or analyses of the qualities of their subclasses or instances.

3.3. Ontological Systems Analysis and Synthesis of "Human"

When we view the world from an ontological perspective we see that it is organized in different levels or categories. This categoric approach was first described by Aristotle and has been used extensively since then. Some contemporary scientists (Gaines 1988) also take the advantage of this analytical approach by seeing the world in categories (also called levels, layers, or strata). Knowledge Level (Newell 1982) is one of the well known articles of this type in AI.

Human nature is not an exception, and analyzed in categories. At first sight, we see human civilizations and cultures; i.e., physical and conceptual human artifacts, which are subject of sociology. As we further analyze the entity human, we see that all his social interactions are solely based on his behavioral nature; in categorical perspective, social organizations are built on psychological one. In further analyses, we recognize that those are based on neurophysiological and physiological organizations. Physiology is organ-level organization of molecular interactions, i.e. biochemical and biophysical phenomena, which are ruled by general laws of physics (see Figure 3.3.).

This is an example for an initial domain analysis through OSAS. It is a very high level analysis of "human." As in all such analyses, it is executed in top-down fashion. The next action is the synthesis closing the feedback loop. It determines whether all components extracted through analysis represent the entire spectrum of human beings. Those are not the whole components we deal with in social and biomedical disciplines. There are many others that can be aggregated into certain

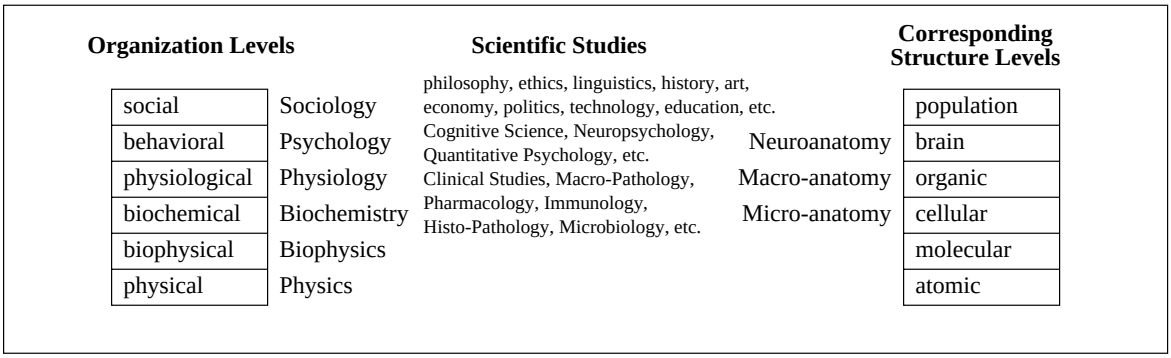


Figure 3.3. Different levels of organizations of human and their corresponding structures and scientific disciplines.

levels of organizations of our analysis. For example, activities and artifacts concerned with philosophy, ethics, linguistics, art, history, economy, politics are aggregated in social level (see Figure 3.3.). All those are subjects of sociology at large.

While we analyze the physiological, biochemical and biophysical categories deeper, we see that each consists of many other disciplines of natural sciences. For physiology there are many biomedical basic sciences such as anatomy (which determines the spatial relationships of those processes), pathological anatomy, pathophysiology, pharmacology etc., and there are numerous clinical fields such as cardiology, neurology, nephrology, ophtalmology, orthopedics and so on. For biochemistry, there are also many biomedical sciences which are classified by the molecular level organization. Some of those sciences are metabolism, endocrinology, immunology, genetics, histology, histo-pathology and so on. Biochemistry is built on the knowledge of inorganic and organic chemistry, and all biochemical events are ruled by physical rules

and laws which are investigated by a single discipline specific to human nature called biophysics. Biophysics does not only consider molecular level physics but also macro level as well, and then is called biomechanics. All branches of biophysics are certainly studied in physics in a lower level, such as in kinematics, physical chemistry and quantum physics.

The process we reported in previous paragraphs is the initial cycle of OSAS. We have analyzed human organizations and constructed a system of sciences based on the relations between those organizations that have been extracted. the former activity is called *Ontological Systems Analysis (OSA)*, where the latter is called *Ontological Systems Synthesis (OSS)*. The synthesized perspective is propagated back to be analyzed so that the feedback loop is closed. After the first iteration, we see that it did not include every activity of the whole biomedical science spectrum. After the second analysis, we have seen that those others are just derivations or components of the ones we illustrated in levels; e.g. most of the clinical sciences are based on the studies of pathophysiology which in turn is based on the physiology. So, the system in biomedical organization level is completed at this highest level.

In this approach, we consider the human being at large. Have all levels of human nature been considered? Since we are computer scientists, we should at least have to ask, "Where is the Computer Science here anyway?" It is not there. In fact, we had not seen that incompleteness before we wrote this down. After the explicit execution of the OSAS, the picture (current synthesis) has become an *object to us*

for criticism exactly as Popper said above. It is now obvious to us that many other things have been neglected. Where is mathematics for example?

There are structural organizations, such as physiological (not physiology as a science it self), biochemical and physical ones. These organizations are factual. But there is no mathematical organization as a part of human structure. It is an organization of human at conceptual level; it is artificial since it is solely a human artifact. After many OSA-OSS loops in our mind we have come up with the following organization levels (see Figure 3.4.).

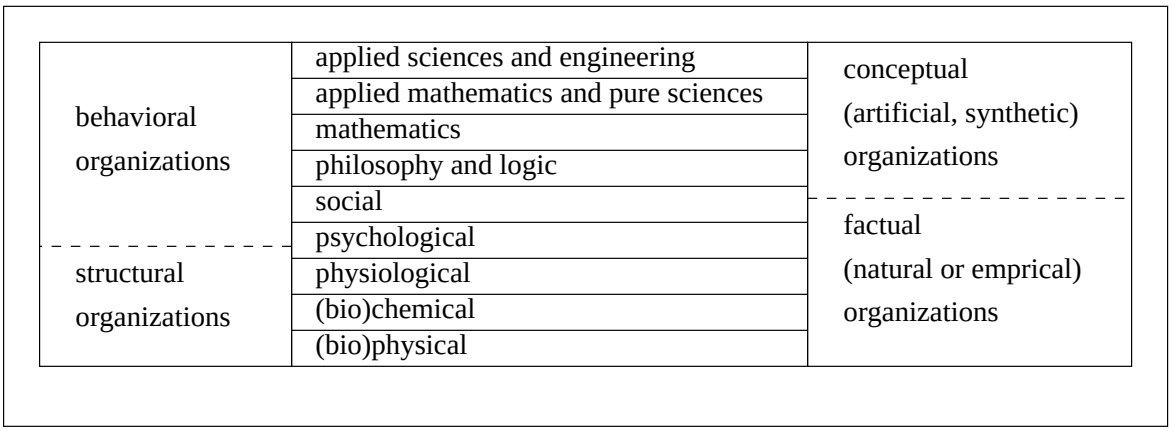


Figure 3.4. Levels of human organizations after Ontological Systems Analysis and Synthesis.

First come physical forces, then atoms, molecules and biomolecules. After organizations of those molecules, organisms come. Animals (nervous system control and psychological organization) appear after the evolution of primitive organisms. There-

after, social organizations are observed in some species, such as bees, ants, gorillas, and so on. Finally, thought and its artifact language comes as human appears.

Each organization is based on the rules and laws of its antecedents. Whenever an organization is built based on an existing organization, it also establishes its own rules and laws (Bunge 1977). Therefore, the higher organizations we deal with, the more often we fail in our understanding of its nature. In order to establish sound rules and find the laws of the nature of higher level organizations, we must be consistent with the rules and laws of the lower level organizations; thus we have to solve problems of lower levels and establish a solid foundation for the higher level organizations. The logical consequence of this observation is we have to start to this enterprise from as low a level as we can, perhaps from the physical level organization. Unfortunately, our medical and computer science backgrounds do not suit that requirement. Therefore, we have decided that the level of biochemistry is the most appropriate one to implement our OSAS approach. Its representation environments, the Multifaceted Ontological Networks, are tested on a biochemical entity, called Citric Acid (or **T**ri**c**arboxylic **A**cid) Cycle, or in short, the TCA Cycle.

3.4. Ontological Systems Analysis and Synthesis at a Single Level

OSA is performed in a top down fashion. Analysis generally is based on decomposing a system into its components, and their individual behaviors are observed for understanding the whole. In OSS, on the other hand, those system components (along

their interrelations) should be put together by establishing proper interrelations between them. This mechanism is executed circularly until the system representation reaches the desired maturity level.

During OSA the entities are decomposed. In the process decomposing the entities in the TCA cycle, we recognized there are only two main types of entities: Structures and Processes. However, structures sometimes define a space which itself is not a structure but a container of some other structures. For example, the cell membrane establishes an intracellular space for all other subcellular structures. Since biological organisms are very dynamic, those substructures change rapidly either in quality or in position, but the limiting structure exist as a frame for much longer period. We call this limited space a *compartment*. In biomedical modeling, the compartment concept has some important properties. It is a means of idealization. Mathematical modeling does not care about the *structure* it self but is interested in the results of the events, which are formulated through some differential functions. In this approach the compartment is considered an *ideal* space in which the molecules (i.e., variables of equations) are evenly distributed. Identifying compartments within our representation may facilitate interactions with biophysical submodels in the future.

3.5. The Citric Acid Cycle Example

Our first prototype implementation of Multifaceted Ontological Networks is the representation of Citric Acid Cycle, a major biochemical process in human metabolism.

The reason to represent a biochemical entity is due its independence from other disciplines except physics and biophysics.

Biochemistry deals with chemical reactions in living organisms, such as human. We must establish relations between these biochemical processes. The most prominent type of relationships are spatial and temporal. These processes occur in micro-spaces. That space is usually cells (intracellular compartments). If we are not able to state where those processes happen, we cannot complete the biochemical world. Therefore, we must not only express some biochemical processes in time but also form a representation of micro-anatomical (or histological) structures in which those processes take place.

In more complex (higher level) organizations, the ontological relations should be more involved. At the biochemical level, all we probably need are three classes of relationships: *Spatial*, *temporal* and *qualitative*.

3.5.1. Spatial relationships

In our analyses, we have distinguished the following relations within the class of spatial relationships:

- componential relations
- compartmental relations
- adjacency relations

Componential relations

These are based on mereological (or theory of whole-parts) (Smith and Mulligan 1982), and set theoretical (class-member) relations. During our analysis of the system, we need to decompose a structure in its substructures and its subclasses thoroughly; therefore we must have following two relations:

- `substructures_of`,
- `instances_of`.

For example, we can decompose `ecosystem` into two classes, `organism`s and `substance`s. The relationship between `ecosystem` and its decompositions is a superstructure-substructure type relationship, and can be represented in `substructures_of` predicate as follows;

```
substructures_of(ecosystem,[organism,substance]).
```

In this type of relationship, the granularity of representation decreases; i.e., the granularity of `ecosystem` is greater than those of `organism` and `substance`. Obviously this is *a* way of decomposition, a single view toward `ecosystem`. Some other views can decompose it differently based on their pragmatical needs. In our (medical) knowledge acquisition tool, *medKAT* users are encouraged to decompose entities in every sensible way. Those different views are managed via facets. Facets are treated in Chapter 5.

Another type of componential relationship is the superclass-subclass relationship. In this type of relationship, the granularity of structures does not change. For example,

```
instances_of(organism:all,
  [complex_organism,primitive_organism]).
```

or,

```
instances_of(country:all,['USA','Canada','Mexico',...]).
```

As seen in these examples a $< subclass >$ is (always) a $< class >$; e.g., USA is a country,[9] or complex organism is an organism. Since the granularity is not changed we can easily substitute class and subclasses with the relevant entities in the template $< subclass > is\ a < class >$.

Any new entity introduced is checked whether it has a connection to the root structure, *ecosystem*. Every entity should be grounded; i.e., every entity should have at least one path to the `ecosystem`, since we know that everything in the world is a part of the `ecosystem`, and we should be consistent with that. This is essential because it is the only way we can know that the newly introduced entity is consistently bound to the representation system and is now a part of it.

[9]A country is obviously not a structure. This example is given to make the distinction between class-members and whole-parts relations explicit.

Compartmental relations

A *compartment* is a space limited by certain structures. For example a room is a compartment limited by a ceiling, a floor and walls. A chair, however, is not a compartment. The compartment is a very useful concept in biochemistry and biophysics. It is an idealization (or approximation), in that it is assumed that certain molecules in a compartment are distributed evenly. This idealization helps the scientist to form mathematical models. We also need compartments in order to distinguish relations like between room and chair. In biology, a cell is a compartment and includes a few types of organelles and many organic molecules. An organelle is a structure with specialized functions, and also forms a compartment where those functions occur. We have distinguished three types of compartmental relations:

- `compartment`,
- `limits_of`, and
- `occurs_in`.

The relationship between a structure and its compartments can be represented in the predicate, `compartment`:

```
compartment(cell, [intracellular_space]).
compartment(mitochondrion,
  [intermembrane_space,mitochondrial_matrix]).
```

The first example implies that a cell is a structure which forms a compartment, called intracellular space. The second one says mitochondrion is a structure, which has two compartments inside.

We can define the relationship between compartment and its limiting structures as follows:

```
limits_of(intracellular_space,
  [plasma_membrane]).


limits_of(intermembrane_space,
  [mitochondrial_outer_membrane,
   mitochondrial_inner_membrane]).
limits_of(mitochondrial_matrix,
  [mitochondrial_inner_membrane]).
```

In these examples, the first argument is a compartment, and the second one is a list of structures which envelop that compartment. The first example implies that the intracellular space is limited by a plasma membrane.

As another example, if we want to express a kitchen in this way we might form our predicates as follows:

```
compartment(kitchen,[kitchen_space]).
instances_of(kitchen,[our_kitchen]).
```

```
limits_of(kitchen_space,
  [kitchen_ceiling,
   kitchen_floor,
   kitchen_wall]).
substructures_of(kitchen,
  [kitchen_ceiling,...,
   kitchen_table,
   kitchen_chair,
   refrigerator,...]).
compartment(refrigerator,
  [freezer_space,
   refrigerator_space]).
```

These predicates imply that kitchen is a structure, which has a compartment called kitchen space; kitchen space is limited by ceiling, floor and kitchen wall; kitchen has many components, one of which is the refrigerator, which has two other compartments, and so on.

The last compartmental relationship we created is `occurs_in`, that relates a process with compartments in which it may occur.

```
occurs_in(tca_cycle,[mitochondrial_matrix]).
```

This example implies that Citric Acid Cycle (also known *Tri*carboxylic *A*cid Cycle) `occurs_in` a compartment, called mitochondrial matrix.

Adjacency relations

We need to establish some more morphological relations other than compartmental ones for expressing where biochemical processes occur. Some processes, for example, occur on certain specialized surfaces and regions of some organelles. Those regions should be described with respect to other regions and the structure where those regions reside. Adjacency relationships are essential for spatial reasoning. Their importance might be appreciated if macro-anatomical relations are considered. For example, every muscle group is attached to certain regions of relevant bones and the arrangements of muscles should be defined in relation with other muscle groups, fasciae, and bones. The same is true for blood vessels and peripheral nerves.

There are four different adjacency relations we define:

- `regions_of`,
- `neighbor_regions`,
- `neighbor_structures`, and
- `neighbor_compartments`.

A structure may have some regions that are called with special names. In order to find some other regions, one may use those for reference. The first argument in `regions_of` is a structure; the second one is a list of regions in/on that structure.

```
regions_of(mitochondrial_inner_membrane,
  [intermembrane_space_side,
   hydrophobic_space,
   matrix_side]).
```

In this example, we see that mitochondrial inner membrane has three regions, which are intermembrane space side, hydrophobic space and matrix side.

The next predicate is `neighbor_regions`. It establishes relationships between the regions. It is basically an adjacency list data structure. It has three arguments that are a structure, a specific region on/in this structure and a list of regions adjacent to that specific region, respectively.

```
neighbor_regions
  (mitochondrial_inner_membrane,
   hydrophobic_space,
   [intermembrane_space_side, matrix_side]).
```

The third adjacency relation is `neighbor_structures`. It is also an adjacency list data structure. It has two arguments. The first one is a structure. The second one is a list of structures adjacent to (or in contact with) the first one.

```
neighbor_structures(mitochondrial_outer_membrane,
   [cytoplasm,
    intermembrane_space_content]).
```

neighbor_compartments is our last predicate for spatial relationships. It relates two adjacent compartments and the structures between them; i.e., it has three arguments. The first and third ones are compartments, the second one is a list of structures, which usually includes a single structure item. If there is an inner/outer relation between compartments, the first one would be the outer compartment and the other would be the inner one.

```
neighbor_compartments
  (intracellular_space,
   [mitochondrial_outer_membrane],
   intermembrane_space).

neighbor_compartments
  (intermembrane_space,
   [mitochondrial_inner_membrane],
   mitochondrial_matrix).
```

3.5.2. Temporal relationships

In our ontological analyses, we conceptualize a process as a function of structures and time. In other words, even if a structure does not change qualitatively (or significantly), we know that processes occur over time. However, these changes may not always be observable because of the limitations of technology in hand. For example, while a chair changes dynamically at the molecular and atomic level, we are not aware of those changes. It is either impossible to observe those changes or would not be practical in most cases.

We have distinguished four types of temporal relationships:

- `process`,
- `followed_by`,
- `subprocesses_of`,
- `instances_of`.

The changes in structures are represented in the predicate called `process`.[10] It has five arguments: The first one is the name of the process; the second one is a list of input structures (we also call these substrates); the third one is a list of output structures (or products); the fourth one is a list of control structures (enzymes in

[10]We do not state *time* in the network representation directly. The time is observed indirectly in processes.

biochemistry, or agent (actor) in general); and the last one is a list of conditions necessary for that process.

```
process(citrate_synthesis,
        [acetyl_CoA, oxaloacetate, 'H2O'],
        [citrate, 'CoASH', 'H+'],
        [citrate_synthase],
        [dG : -9.0 - -9.2]).
```

In this example predicate process implies that process citrate synthesis takes three substrates (Acetyl Coenzyme A, Oxaloacetate, and a molecule of water) and produces a molecule Citrate, Coenzyme A, and a Hydrogen radical. This reaction is catalyzed by the enzyme Citrate Synthase. The total of free energy is required to be between 9.0 – 9.2kcal.

In a complete representation system the next (following) reaction in a chain of reactions can be predicted by searching the input lists of processes and comparing the contents of those lists with the molecules and elements available in that location of the system after a certain reaction; however, this search might be very costly in a large set of reactions. Therefore, we also proposed another predicate, called `followed_by`. It is not an essential predicate, but makes computation easier. This predicate has two arguments. The order of the arguments indicates also the order of processes.

Generally, at the biochemical level the first process produces some molecules which are essential for the second process.

```
followed_by(citrate_synthesis, isocitrate_synthesis).
```

The third temporal relationship predicate is `subprocesses_of`. This is very similar to the spatial relation `substructures_of`. This one changes the granularity of time instead of the granularity of structures. We consider that every process occurs in a time interval. When this interval is decomposed into time intervals, then processes are reduced into subprocesses. After awhile, we may not be able to discern further subprocesses by decrementing the granularity level; however, even if the changes are not observable, we can still potentially capture them within our representation.

In the following example, we see the subprocesses of Citric Acid (or TCA) Cycle.

```
subprocesses_of(tca_cycle,
                [citrate_synthesis,
                 isocitrate_synthesis,
                 isocitrate_dehydrogenization,
                 alpha_ketoglutarate_dehydrogenization,
                 succinyl_CoA_synthesis,
                 succinate_dehydrogenization,
                 fumarate_hydratization,
                 malate_dehydrogenization]).
```

The last relationship is `instances_of`, which we have analyzed in spatial relationships. There is no difference between these two `instances_of`s. Both indicate members of a specific class of entity. In the context of temporal relationships, we can analyze this as follows: Citrate synthesis reaction, for example, has certain properties. If we want to duplicate the same reaction in laboratory we would observe *specific* qualities in our reaction instead of a range of qualities of that reaction class. That is, we are gathering *samples* of a process, which must fit the temporal model of that process. For example,

```
instances_of(citrate_synthesis,
             [my_experiment1,
              my_experiment2]).
```

In this example, `my_experiment1`, and `my_experiment2` are citrate synthesis reactions, i.e., processes, we observed in laboratory in certain different days. The data we collected should be coherent with the data specified within the citrate synthesis process class.

3.5.3. Qualitative relations

Although qualitative relations can be acquired for every single entity through ontological analysis, there is no fixed set of qualitative relations. In our analysis we acquired many qualitative properties for certain entities. Some of those qualities are familiar and used in daily life, such as weight, length, volume, quantity, color etc.

Some of them are specific to a particular type of entity (and therefore difficult to communicate), such as molecular weight, iso-electric point, electro-negativity and so on. Some others are highly vague or difficult to describe, such as "shape" (Davis 1990; McDermott 1987).

In Chapter 4.6., we mention that the qualitative criteria of classifications should be made explicitly. These are the most important qualitative relations with respect to ontological significance. Assume that we have a list of letters, called `letter_list`.

```
letter_list [A,b,C,d,e,F,G,h]
```

This list can be decomposed in many ways, two of which are as follows:

```
letter_sublist_1 [A,C,F,G]
letter_sublist_2 [b,d,e,h]
```

based on the uppercase-lowercase relations, or

```
letter_sublist_1 [A,e]
letter_sublist_2 [b,C,d,F,G,h]
```

based on the vowel-consonant relations. As also seen in this example, we may classify things based on their certain qualities. In a rigorous representation, we must make the classification criterion (i.e., referenced quality) explicit.

As we have seen in examples at the beginning of this section, qualities can also be numerical.[11] The spectrum of quality representation is highly broad, from very fuzzy qualities (such as cold) to highly precise ones (such as 0 Kelvin). The relations between fuzzy and precise extrema should be established for each represented quality so that everyone can elicit the same semantics from the representation of data.

[11]The word qualitative is usually used in contrast with the word quantitative which is also used with the term "numerical" interchangeably.

CHAPTER 4

ONTOLOGICAL NETWORKS

An Ontological Network is a representation platform for formal knowledge extracted through OSAS. It is implemented in Logic Programming, i.e., in formal logic. However, it also follows the representational formalisms of semantic networks and frame representations. This work will not provide a formal introduction of those types of representation; however, a short review of their properties important to Ontological Networks is provided.

4.1. Semantic Networks

Semantic Networks are knowledge representation tools in graphical formats. Though the program level representation can be accomplished by any type of approach (either procedural or declarative), the more logical way of doing this seems to be the declarative approach. In particular, Semantic Networks are nothing but logical relations between argument tuples illustrated as graphs. Therefore, logic programming is a natural representation tool for Semantic Networks. Many authors recognize that there is no substantial difference between the representations in predicate logic and in semantic networks(Cercone and McCalla 1987).

We think that in terms of expressiveness, the semantic network formalism is a subset of formal logic and sufficient one for most of the cases. It is usually restricted via certain formalism by diverting from the amorphous predicate logic. Inheritance is perhaps the most important feature among those. In semantic networks, the inheritance relation is made explicit along the lines of *is-a* relations. In first-order predicate logic `isa` can be any type of relation without that formalism.[12] In the context of knowledge representation, formal logic and semantic networks are analogous to the relationship between assembly language and high-level programming languages. The former is more powerful to express but less efficient to understand and to work with. The latter is a more useful for feedback and might also be seen as "flow"-charts of declarative programming.[13]

Our principle in this work is making everything as explicit as possible. For example, we do not need to represent every integer multiplication *procedure* explicitly but express the procedure how to execute.

---

[12]Namely, `isa` is converted into `instances_of` relationship in our implementation at the logical representation level, because we need to state the most basic type of an entity as a structure, process, compartment, or just a degree of a quality with the predicate `isa`. It semantically suits itself best to this most primitive type-relation. We represent the higher order type-relations between classes and subclasses in `instances_of` relation.

[13]But some of the relations, predicates with more than two arguments, cannot be thoroughly illustrated within semantic network style graph representation.

There are two types of implicit knowledge essential to consider: 1. Metaknowledge for reasoning and problem solving, and 2. Metaknowledge for conversion of data from one format (or representation) to another.

The first type of metaknowledge consists of logical reasoning mechanisms, software systems attached to Ontological Network for certain problem solving. This type is beyond the scope of this work. In this thesis, we solely consider how we can represent factual (natural) knowledge without (or with minimum) design commitments.[14] The failure of conventional knowledge representation is also caused by those commitments. Represented knowledge is essential for reasoning (and any type of metaknowledge) but not *vice versa*. Our aim is to represent knowledge sharable by all reasoning systems via specific facets. Therefore, we do not want to represent intentional knowledge in this context.

The second type of metaknowledge consists functions converting data from one format to another. For example, consider a set of data collected via signals of a computer tomography scanning a human brain. We may convert it into several two dimensional visual slices or into three dimensional images of certain parts of the brain (e.g., ventricles). This engineering knowledge is not a kind of knowledge we aim to represent in our medical knowledge base, but we need to *use* that meta-knowledge for our practical purposes. We do not represent it, but instead use it as a tool. It has to

[14]Some authors instead call this ontological commitment (Gruber 1993; Davis, Shrobe, and Szolovits 1993).

be attached via facets (an interface) to our system as satellite software (reasoning) system.

Semantic networks are represented in directed graphs whose nodes are labeled with concepts of any kind (without any restriction) and edges (arcs) are labeled with the relation between two nodes. It is a better visualization and abstraction tool for represented knowledge when compared to the *lines* of formal logic (see Figure 4.1.).

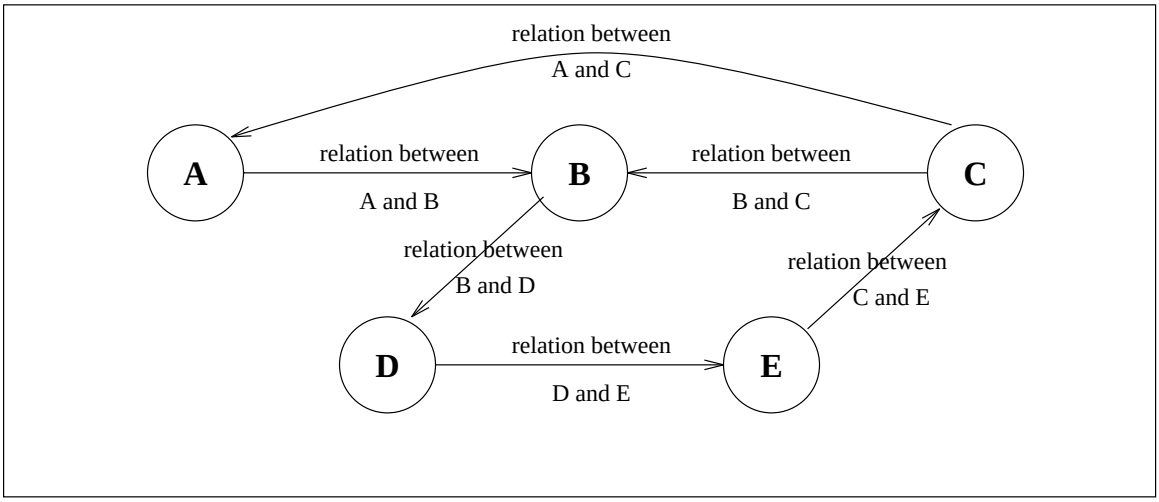


Figure 4.1. A semantic network representation convention.

However, the expressive power of formal logic is greater than that of semantic networks.

4.2. Predicate Logic

Even if we restrict our selves in Prolog convention rather than Logic Programming at large, we can represent many arguments in a Prolog line (i.e., in a predicate) in various forms such as atoms (not necessarily just two atoms as in semantic networks),

```
relation(a,b,c).
```

lists,

```
relation([a,b,c,],[d,e,f]).
```

other predicates (in higher-order logic),

```
relation1(relation2(relation3(a,b),c),d,e).
```

or any combination of these.

```
relation1(a,[b,c],relation2([d,e],[f,g])).
```

However, in semantic networks there is a single format: Two nodes and a directed edge between them. Sometimes, it is better to have a list of items, like a check list which we use in daily life. Its corresponding representation in semantic networks may look awkward.

4.3. Frames

This list type representation is utilized in *frames*. Frames are essentially no different from semantic networks, but are considered more intuitive because of their format (see Figure 4.2.). Because of their format and their imposition of hierarchical

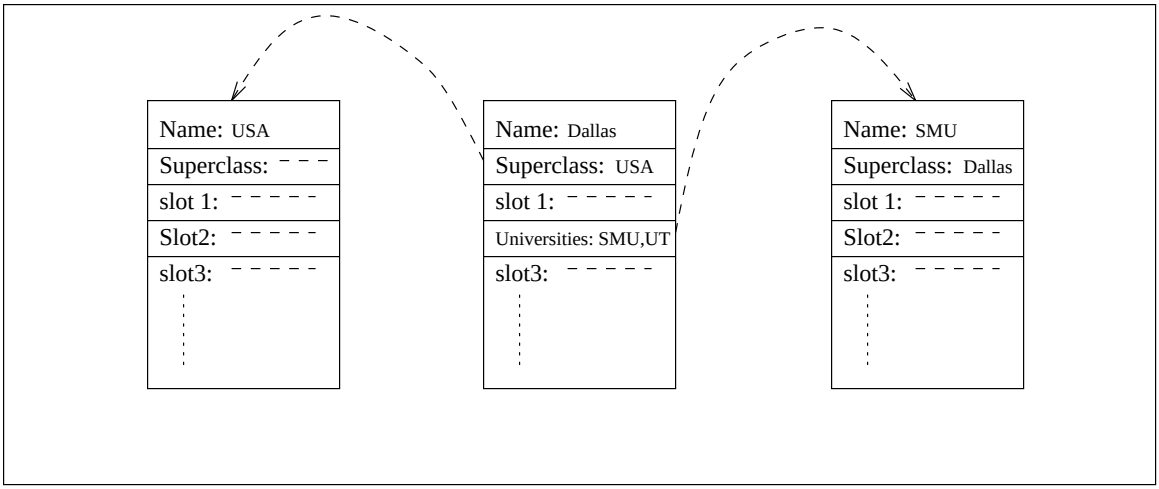


Figure 4.2. Frames are organized hierarchically.

organization, they also are called structured representations. A frame consists of a list of *slots* which are used to represent properties of a class or object. Slots can be of any type. They can refer to another frame, as in Figure 4.2., or can be a value.

Mechanisms for inheritance in frames are not well documented, and there is no significant distinction between frames and semantic networks in this regard. Entities inherit their qualities from their parent nodes or superclasses unless they explicitly

specify their own qualities (i.e., unless they overwrite those properties). A modified version of the classical "bird" example can be stated as follows (see Figure 4.3.).

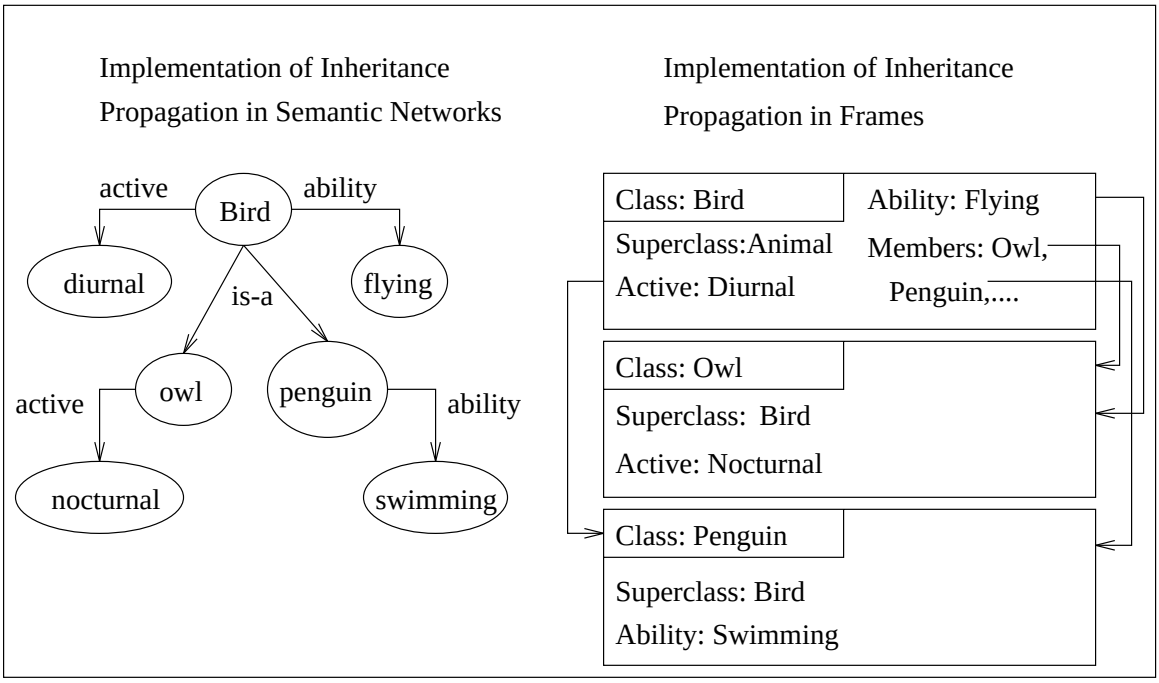


Figure 4.3. Inheritance propagation in semantic networks and frames.

Since owls and penguins do not share common features of birds, they need to overwrite some specific properties that birds usually have, i.e., owls are nocturnal and penguins are flightless. However, this method of representation results in non-monotonicity in terms of the dependency in inheritance and contradictions between entities on the same path of the tree. This is called nonmonotonic justification(Barr and Feigenbaum 1982). The reasoning mechanism is called *nonmonotonic reason-*

ing, a current branch of contemporary logic. It is not a mature area[15] of logic. This method of representation is a very significant design commitment and make the power of well studied classical formal logic unusable.

4.4. Ontological Networks as a Knowledge Representation Platform

We have seen that no concept or entity has meaning in isolation. The meaning of an entity is acquired through the aggregations of some (sub)entities and the relations between that entity and other entities. Those relationships can be suitably represented in a (semantic) network illustration. Although we use Logic as a representation medium, we need to establish some high-level methodologies, part of which have already been accomplished by existing representation formalisms such as semantic networks and frames. Therefore, we will borrow some of their methodological properties when necessary, while representing knowledge in formal logic at software level so that we can fully use the power of all these three conventions.

The differences between the Ontological Network and Semantic Network con-

[15]Some examples regarding their failures may be given for every modern approach in logic though, we here would like to cite just an instance for one of the most popular ones among all, namely circumscription (McCarthy 1985; McCarthy 1980). "It isn't clear whether there is a problem with the systems of nonmonotonic reasoning or whether we simply don't yet have the right axiom sets. Anyway the existing formalizations don't have enough of what I call *elaboration tolerance*. The idea is that formalizations should follow human fact representation in being modifiable primarily by extension rather than by replacement of axioms" (McCarthy 1993). "While circumscription in some cases provides a simple and efficient way of describing minimality, there is no getting around the fact that it must fail in others" (Jaeger 1993).

ventions as well as the reason to include frames in the representation is elaborated in the following sections of this chapter.

4.5. Methodology Driven versus ad hoc Extractions of Relations

In Ontological Networks we represent entities and their interrelations in nodes and labeled directed edges, as in the semantic network convention. Numerical and nonnumerical qualities of entities are represented in Frames. Their values are stored in slots of frames. Data (quality value of an entity) is stored in the frame attached to the entity node or can be accessed through the frame address stored in that frame.

The structures of Semantic Networks are not based on analytical methodology; therefore, the established relationships are *ad hoc* and may be very numerous. In a reasonably sized knowledge base types of control such as consistency and completeness are not easy to maintain and update when there are so many relationships. On the other hand, in Ontological Networks the established relationships are as essential as possible. The "minimal vocabulary" and factual nature leads to a systematically organized representation. If those relations are not well selected, as may happen naturally, they are refined progressively through OSAS so that the most essential relationships between the fine granularities of entities are revealed. Since the representation is based on such basic relations, the monotonicity of the building blocks throughout the Ontological Network is maintained. That results in ease of maintenance, in addition to reusability and sharability of knowledge.

4.5.1. Consistency checking of the relationships

Consistency checking can be handled much easier than a network with many *implicitly* overlapping and overloaded relationships. For sake of the expressiveness, using highly overloaded (many times as compound phrases stated) relations is a common mistakes in knowledge engineering (Lenat and Guha 1990). That unfortunate approach helps to aggregate many concepts into a single knowledge module, which may be clear to the knowledge engineer with his background in natural language but is a mystery for the computer. In Ontological Networks, such mistakes are eliminated through the refinement process of OSAS.

4.5.2. Monotonicity of the knowledge

These properties let us build a systematically structured network and implement a "monotonic" inheritance that is consistent with formal classical logic and has minimal design commitments (see Section 4.9.). Classical logic has minimal commitment and builds a base for the modern forms of logic. We even refrain form some of its commitments with respect to the quantifiers, namely $\forall$ and $\exists$, in this work. However, through appropriate tools such as fuzzy and probabilistic conversion functions, it is possible to come out with more realistic quantifiers representable in the quality sets of entities. "Heavy" design commitments such as "exceptions" in nonmonotonic representations prevent their knowledge systems from sharing knowledge with others.

Representations based on their reasoning approaches, such as nonmonotonic reasoning, are incompatible with other systems which use other reasoning mechanisms. Our principle in this work is establishing minimal design commitments in the representation, in order to interact with any system while communicating with reasoning tools via facets on which the represented knowledge can be modified (if necessary) according to the specific needs of reasoning systems. Our declaratively organized Ontological Networks should interact with satellite reasoning systems via facets since those systems are organized procedurally.[16] Ontological Networks must be purely declarative by containing factual knowledge, representing all facts we can provide. This approach, basing the reasoning mechanisms on powerful extensions, is also suggested by McCarthy as "[Reasoning] formalizations should follow human fact representation in being modifiable primarily by *extension* rather than by replacement of axioms"[17] (McCarthy 1993).

4.5.3. Detecting gaps in the knowledge

Ontological Networks also make any lack of the knowledge explicit. If knowledge does not exist, concepts cannot be decomposed or related to other concepts. The state of knowledge in an Ontological Network is mostly at the granularity level of leaves

[16](Since) reasoning *is* procedural (Davis and Buchanan 1985; Russell and Wefald 1991) even in case it is applied in declarative style.

[17]Emphasis is added. A longer version of this quotation can be seen in Page 68 as a footnote.

in graph and relations established between nodes of entities. Knowledge regarding the entity qualities is hidden in invisible frame hierarchy however (see the next three sections). If we are aware about what we do not know, then we can direct our research efforts toward that problem effectively.

4.6. Data Representation in Ontological Networks

In Ontological Networks, strategic knowledge is visible but not the data. The goal of the Ontological Network is to integrate data with the knowledge in our conceptual universe.

This data is massively stored in various database locations throughout the world. It would not make much sense if we proposed to copy all of them into a single representation platform (even if it were possible). It is the concepts behind the data that we must integrate; therefore, we do not merge any data relating other entities with our entities, but keep the complete list of data types for every entity in frames attached to the entity nodes. Through the references in the frame slots we can reach the relevant data.

4.7. Descriptive Numerical Quality Representation

Data is knowledge describing the quality of an entity in some numerical format. *Quality*, on the other hand, is a more general term embodying both numerical and nonnumerical qualities. Instead of saying "nonnumerical" quality, we sometime

use the term "descriptive" quality. Many authors use the term qualitative instead of nonnumerical or descriptive, but it may confuse readers since both numerical and nonnumerical terms describe qualities of entities. Besides counting materials in integer format, such as 1 apple or 2 eyes, numerical representations regarding natural qualities of entities are almost always associated to unit systems that are agreed by everyone, such as 1 meter or 1 inch. In other words, a numerical format is based on a fixed reference commonly and formally agreed upon.[18]

4.7.1. Fuzzy qualities

Descriptive qualities, such as yellow, dark, thick or heavy, on the other hand, are not based on such a solid reference point but rather based on fuzzy references. As a consequence of their ambiguity, the usage of those descriptions may be conceptualized differently by different people.

The distinction between fuzzy and precise is not very solid (consider the history

---

[18] This can be best understood if the historical background of unit systems, in particular of meter, is considered. *"When the metric system was standardized careful measurements were made of the polar quadrant of the earth through Paris, and the metre was originally defined as one ten millionth part of this arc. To overcome the impracticality of the definition the French Academy of Sciences used a platinum bar whose length was equal to the theoretical length, and this (despite an imperfection in its calculation) was used for the next 90 years. In 1889 the international Metre was redefined as the distance between two lines engraved on an alloy bar, and this held until 1960 when the metre was defined in terms of the wavelength of the krypton-86 atom; it was redefined again in 1983 (...) as the length of the path travelled by light in a vacuum during a time interval of 1/299 792 458 of a second."* (Hawkins and Allen 1991)

of the metric unit system) . The exactness or the degree of precision in nature has no limits. The more advanced the technology, the more accurate the definitions of units as well as measurements performed with those units are. From this point of view, fuzziness or preciseness is just a "fuzzy" degree of a measurement itself. Less precise data or more fuzzy qualities can always be acquired by rounding or approximating the values or enlarging accepted intervals of descriptive qualities. For example, if the spectrum is in seven colors only, then turquoise should be either classified into blue or into green. If demarcation lines between all conceptualized colors are not stated formally, two different people may not agree whether to put turquoise in blue or green. However, whenever that line is established formally then even two slightly different turquoise colors can be doubtlessly classified in different partitions of the spectrum.

4.7.2. Representation format of the data

The representation format of data is another criterion to be clarified. The same data can be formulated in different formats. Consider a data sample regarding a certain quality of a population; This knowledge can be represented by providing those qualities as they are acquired, or they can be statistically processed so that knowledge can be represented in terms of a distribution function.

The format of data and its preciseness can be altered in parallel. In the previous example, the property of population could be stated in terms of an average instead

of its function. In this case, the precision would be degraded when the format is changed.

4.7.3. Relationships between data

If there are two qualities, such as weight and height, then the distribution function should be three dimensional (weight, height, and number of the samples at each numerical data pair) in order to avoid diminishing the precision. It still is tractable for us to visualize this 3-D function. If, however, age is also a variable, then the visualization of the function would be very hard. In fact, there could be tens of different qualities for a single entity (if not hundreds or even thousands). The ability to receive data in a desired format from a multidimensional distribution function should be considered an important criterion in a reasonable knowledge representation system for natural sciences, since there is no entity in Nature that is independent from others.

An ideal reasoning mechanism which is designed to predict the changes to a system that result from certain alterations (e.g., in case of therapy scheduling) must utilize these multidimensional relationships between the entity qualities. For this reason we should be able to represent that knowledge by integrating relevant data collections successfully. A complete knowledge system (reasoning systems along with other software packages such as one for statistical analysis, connected to an Ontological Network via facets) ought to process the raw data, simulate the scenario desired

(e.g., administering a therapy), deduce the probable outcomes based on the knowledge represented on the Ontological Network, and provide it to the user in the format and precision he desires, via a user-specific facet.

4.8. Quality Representation in Frames

At the implementation level, semantic networks and frames do not differ, as both can be represented in terms of predicate logic. Conceptually, we illustrate knowledge in directed graphs and frames only for making the relations and distinctions clearer to the user.

There are a number of reasons to use frames instead of semantic networks at the user's level of visibility. We represent knowledge in a graphical convention because it illustrates the conceptual dependencies between entities, but there are no conceptual dependencies between a *quality* of an entity and another entity. The only exception is inheritance between a class of entities and its subclass.

Those frames can help hide the underlying data of the representation. The amount of data is often enormous, and therefore would not be practical to represent it in nodes of a network, even if it were understandable by the user. The amount of data would hide the conceptual dependency relations which are the main purpose of the ontological network.

4.8.1. Representing different granularities in hierarchical frames

We can *virtually* attach such a frame of qualities to each entity node in an Ontological Network. Whenever the quality of a specific node is necessary to see, the user can look it up by activating the frame of a specific node (see Figure 4.4.).

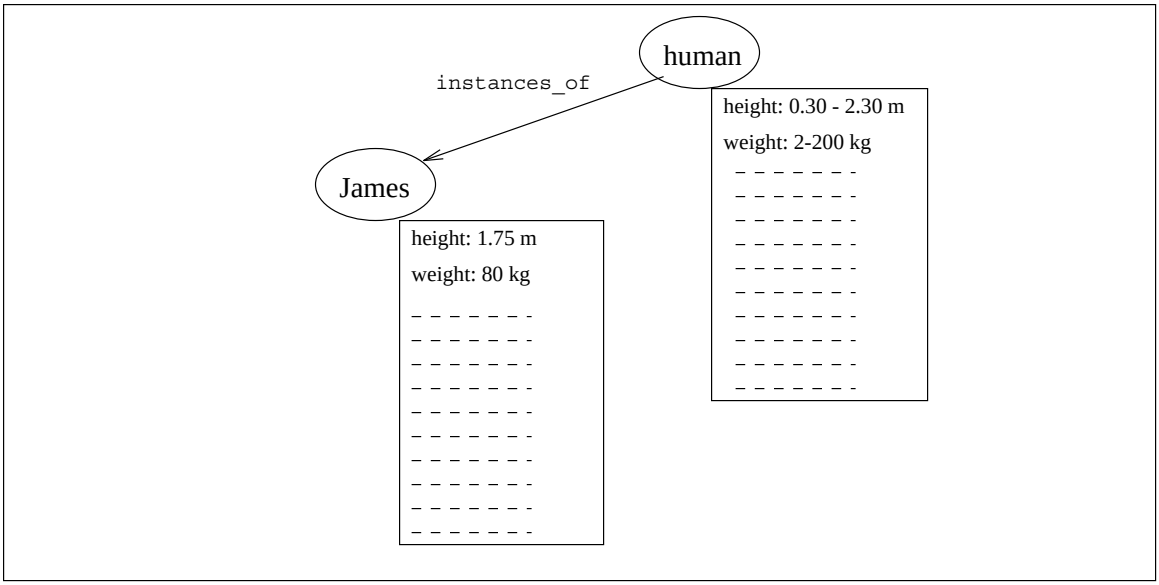


Figure 4.4. Visualization of qualities through frames attached to nodes.

The most important reason, however, goes beyond the visualization convention. The data may be retrieved in certain aggregation steps, each of which can be intuitively represented in the frame format. That is, different users may require knowledge at different levels of granularity, and this is easily supported by the hierarchical nature of the frame representation. For example, assume that there is a highly detailed record (or frame) of qualities for every person of a group in a database. A medical

doctor whose patients have frames in that database may want to see the heart disease risk index of a certain patient. Also assume that there is an established formal method for calculating the heart disease risk index automatically based on the information provided in that database. One of the criteria for this index is the degree of smoking of the patient. However, it is a compound function of the years of smoking, types of the smoking in each year, frequency of smoking associated with those types, the degree of second hand smoke and so on (see Figure 4.5.). Since each of them is a vari-

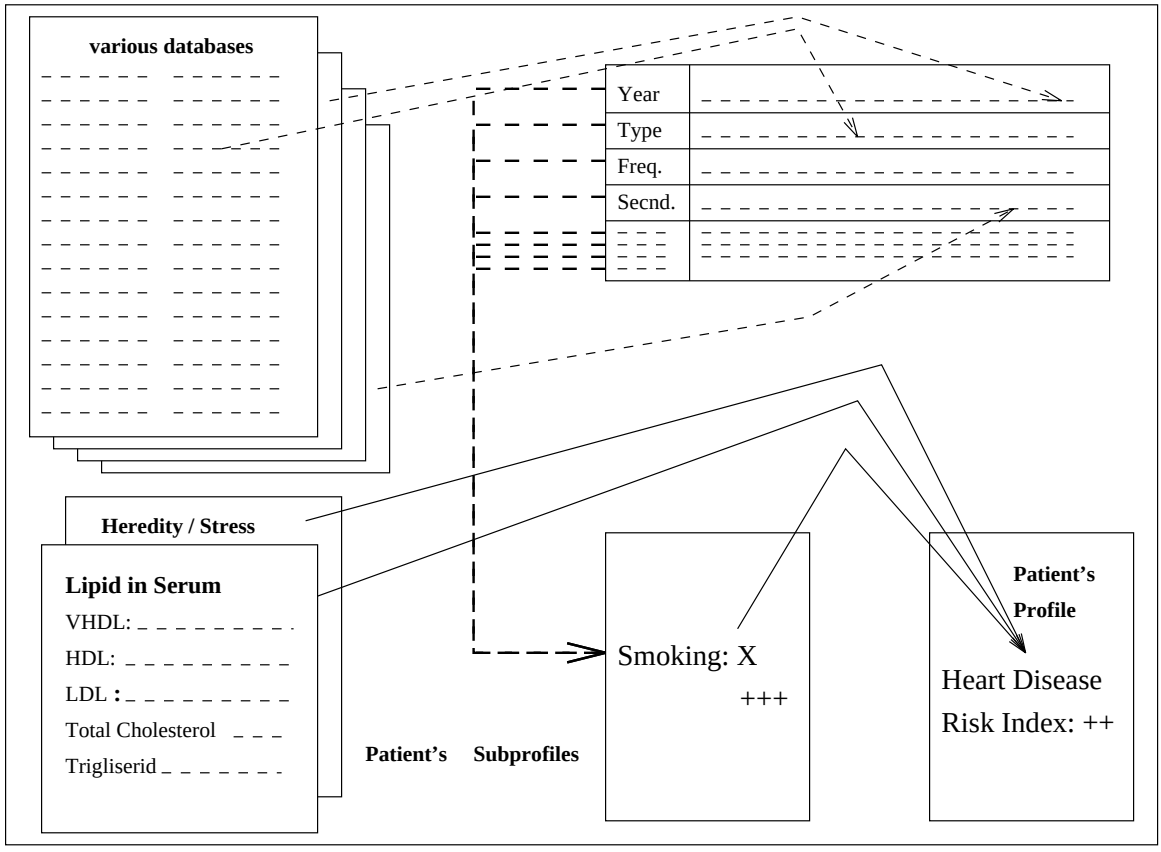


Figure 4.5. Every attribute of a frame is a compound function variable values of which are acquired from attributes of another frame.

able of the function, computing that multivariable function for acquiring the degree

of smoking of the patient is not an easy task for a physician. Today, this risk index is determined by an internist or a cardiologist. Obviously, his decision on the degree of smoking is a highly informal process which varies from physician to physician and in practice, this variation has little influence on diagnosis and therapy. If, however, the computer is a part of the decision process, we would expect it to base its decisions not only on this particular case, but on correlations between complex variables of a patient population as a whole in a large database. When the data that is acquired and represented is retrospectively processed in a scientific research, the importance of its accuracy will be paramount.

As shown in Figure 4.5., the physician is interested in the Heart Disease Risk Index; however, he might as well be interested in the other variables such as "smoking." In the previous function where the index is computed, smoking might be represented, for example, as a vector, which is an undesired format for a physician. He would rather prefer some integer numbers or some plus signs (as shown in Figure 4.5.), a standard estimate of the degree of the smoking. Therefore, both formats must be available in background frame representations, or an idealization function converting that vector to plus signs should be kept in that frame slot so that the number of pluses be computed from a given integer value.

Another example can be taken from meteorology. Meteorological data is stored in databases distributed around the world, and is collected and stored at hourly intervals. Some meteorologists may need that hourly data for weather forecasting or

for investigating a certain important phenomenon which happened on a specific day. Some others may need that data in a yearly format in order to perform research on the climate changes over a period of decades. Those different levels of data may be represented in a frame hierarchy (see Figure 4.6.). If daily data is needed it can be

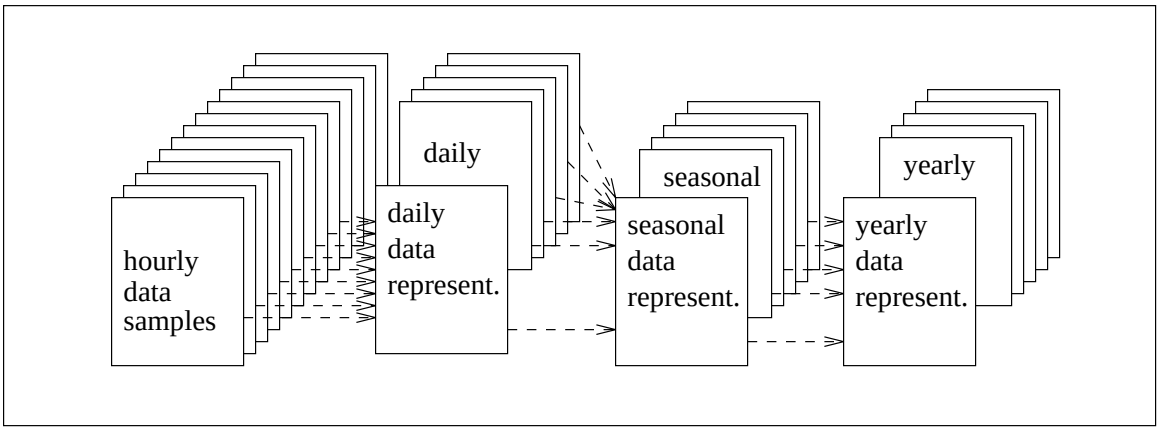


Figure 4.6. Every representation unit at a higher level of hierarchy is a compound representation of a collection of units in its immediate lower level.

produced based on hourly data. If a weekly or monthly format is desired, it can be formed using the daily data and so on.

The Ontological Network is illustrated as many interconnected nodes and data frames attached to those nodes. More specific frames in turn can be reached from the root frame attached to a particular node. The qualities in different slots might be unrelated, but together shape the distinguishing characteristics of the entity. The values in slots may also differ in range from raw data to different types of processed statistical data. For example, we might have a representation of a Texan as

```
%% a classification based on "citizenship" quality

instances_of(human:citizenship:all,

  ['Canadian', 'UScitizen',...]).

instances_of('UScitizen':all,

  ['Texan','New-Yorker',...]).
```

In this example, `Texan` is an entity that represents a member of the class Texans. It has many qualities attached, some of which are unrelated to some others. Those qualities can be depicted in a frame hierarchy (see Figure 4.7.).

The heart disease rate given in the root frame is an overall score. Medical professionals who need more specific data about that type of quality may look that up directly. For example, a cardiologist in Dallas who is examining his patient may want to see the biostatistical distribution of the relevant population, such as Texans between 50-60 years of age with a heavy smoking habit, and whose both parents died from heart attacks.

4.9. Inheritance in Ontological Networks

The inheritance system in Ontological Networks differs significantly from mainstream knowledge representation techniques.

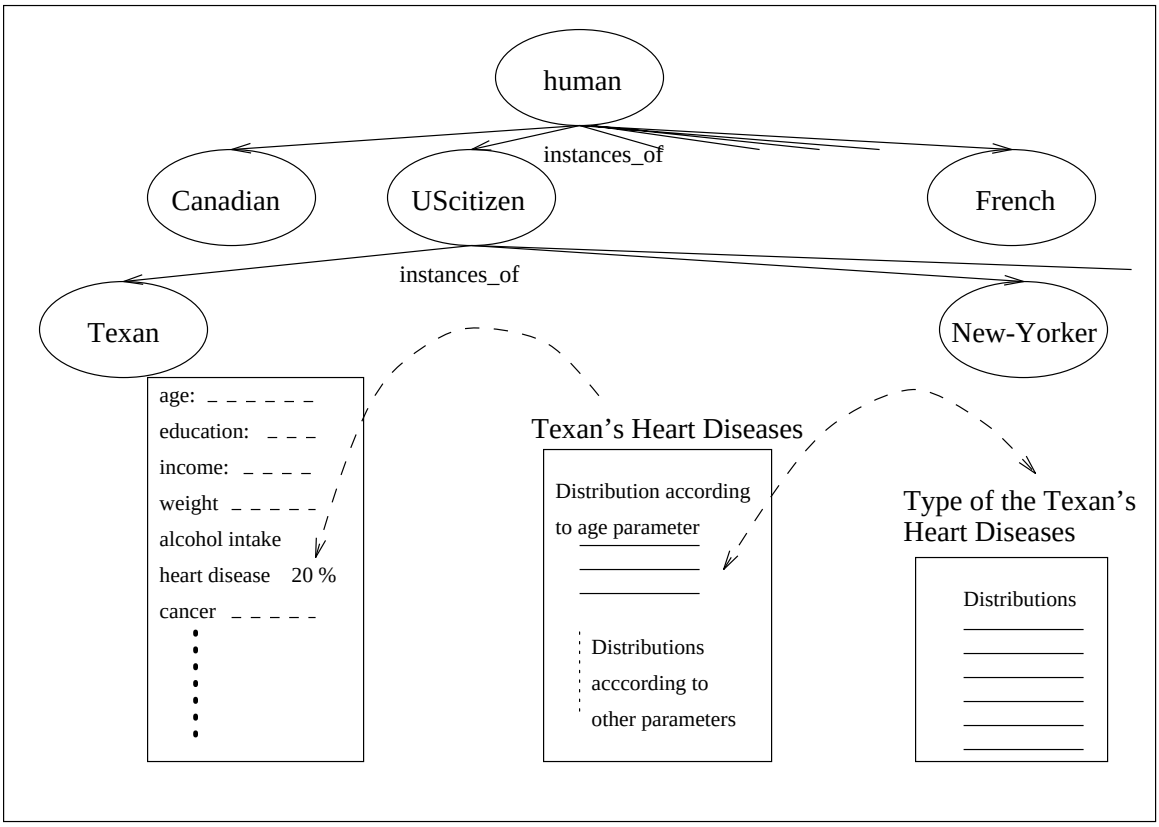


Figure 4.7. Acquiring the relevant format of quality (data) and the desired degree of precision by frame representations.

4.9.1. A monotonic model of inheritance

Through Ontological Systems Analysis and Synthesis, we can build a representation system from simplest building blocks (essential relations) which results a monotonic representation which consists of neither complex nor overloaded nor inconsistent relations. The entities are highly connected to each other, where the degree of

connectedness is a function of the degree of existing knowledge; therefore, that degree would increase for those entities as knowledge increases over time.

In our representation there is no concept of "exception." The quality set of a class is *minimally defined*; i.e., there is no defined quality in a class that is not fully inherited by all subclasses. This type of default quality dependence does not require the user to explicitly represent what qualities are inherited from a class by each of its subclasses.

Since the entities must be consistent we cannot build the famous "bird" example (see Figure 4.3. in Page 67) as it is. In that example, the subclasses overwrite qualities of their superclasses; that is, the quality does *not* hold for all members of the superclass. There are inconsistencies between parent and offspring nodes in terms of inherited qualities.

We have built a Knowledge Acquisition Tool for automating knowledge input for practical purposes. It also partially embodies OSAS principles in that it does not allow the knowledge engineer to enter knowledge which is not consistent with any parent nodes connected through `instances_of` relations. Currently, this relies on the confirmation of knowledge engineer. On the other hand, it should not be very difficult to build a consistency checker in the future that compares the qualities of the offspring nodes with those of their immediate parent (superclass) nodes.[19]

[19]There may be more than one parent, since there is no restriction for multiple inheritance as far as it is consistent in this context.

4.9.2. Classification and inheritance in the natural sciences

Based on this principle, we can build this "bird" example as follows (see Figure 4.8.). Every instance should be consistent with qualifications of its parent. Since

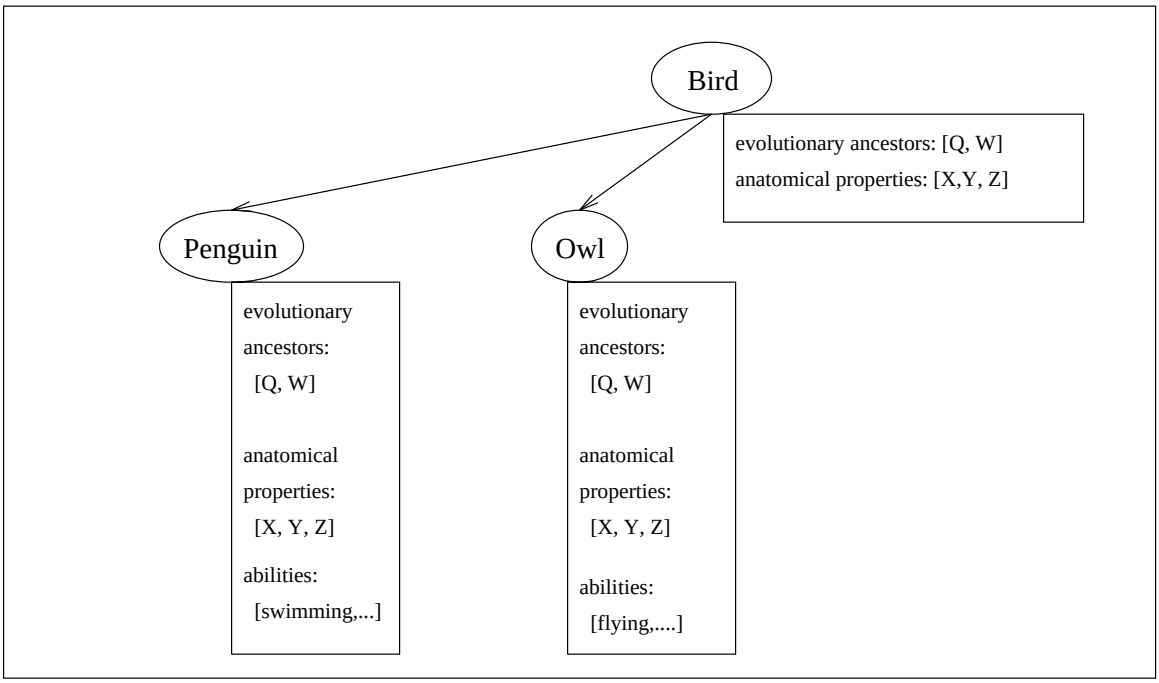


Figure 4.8. There is no inconsistent inheritance propagation from a class to its subclasses.

they are default qualities, there is no reason to state ancestral and anatomical properties again at the instances (subclasses); however, we have restated them above for clarity.

One of the main reasons we take this approach, involves how the field of Biology or Paleontology[20] answers questions, such as "Is X a bird?" As far as we know, the answer is based on the anatomical characteristics of birds and those of X. It surely is not based on only whether it flies or not. We know mammals, such as the bat, which fly, or others which swim such as whales. So, their superficial appearances and simple acts are not bases of cladistics.[21]. In another discipline or from another point of view they could be as well classified into separate classes, such as flying animals (bat, canary, butterfly, ...) and flightless animals (whale, penguin, ostrich, kiwi, ...) instead of birds, mammals and so on. In the natural sciences, subclasses are defined in terms of certain qualities if those qualities are determining factors throughout those subclasses. Obviously, there is no *exception* in biological sciences in this context. If this analysis is performed as much as it deserves than the synthesis is followed immediately: "A bird is an animal, which has X, Y, Z inborn properties in its anatomy, and whose ancestors were Q and W in evolutionary history; therefore, *any* animal which does not fit within this rule is not a bird." If this rule is broken even once, it is not a scientific rule anymore.

As with other natural sciences, the science of biology changes continuously. Every day we learn new things about the nature of the organisms on our planet. Through discoveries based on genetic maps, we may learn sometime in the future

---

[20]Palaios (Greek): Ancient.

[21]Cladistics: A system of classification based on the phylogenetic relationships and evolutionary history of groups of organisms (Houghton Mifflin 1992)

that bird A is not a bird but something else, because, due to some genetic fact yet to be revealed, A's ancestors may not be the same as of other birds. We may also find a bird B (whose ancestors are exactly same as those of other birds) which does not have all anatomical properties a bird has to have. In those instances, scientists would have to have change their rules as well as their Ontological Networks. Even if those scientists do not use and change the Ontological Network we here propose, they still will have to update the Ontological Networks in their mind, which in some way will affect the taxonomic tree used.

As seen in this example (see Figure 4.8.), our representation is consistent with default reasoning, whose author states "aside from mathematics and the physical sciences, most of what we know about the world has associated exceptions and caveats" (Reiter 1985). Since physical sciences are generally considered sciences about inanimate objects (Hawkins and Allen 1991), he may not agree with us about our approach, which involved living organisms. We think that this should work throughout all disciplines, *if* classifying rules are established correctly. If they are not correct or consistent, then we could talk about neither *disciplines* nor sciences. Obviously, if those classifications are established by people who are not knowledgeable about the topic, the result would be like the "bird" example, which is inconsistent in classical logic as it is "shown" by Reiter and many others.

This canonical example has been established in order to show how qualities are passed from a class to some but not all of its members, and to claim that non-

monotonic reasoning is necessary. We think that might be necessary in behavioral and social sciences, but is not necessary in natural sciences. This example shows us that we need to find the essential characteristics of classes and establish only those as qualities in our representation. That is, we believe that classes and qualities should be interrelated, and should not be considered independently. Of course, individual members are not necessarily restricted in this fashion since their characteristics are not inherited to any other entity.

The inherited characteristics here are anatomical qualities that let most birds fly, whereas there are birds that have additional qualities which also affect the anatomy of these birds and prevent them form flying. This analysis shows us that the characteristics are not always coherent among themselves and not equally dominant overall. They sometimes support but some other times they may restrict each other. Therefore, in future we will have to represent qualities in a fuzzy range from prominent to rudimentary. If we can build proper ontological models of processes such as flying, then we must check whether functional qualities such as flying can be performed by an entity while maintaining other qualities such as anatomical and spatial conditions intact. These issues should be studied further to find out the proper quality representation.

4.9.3. Retrospective inheritance

If we consider that "flying" is a quality most of the birds share and wish to relate birds closely with flying, we can establish such a class using the ideas described above. In that case, our famous "bird" example may be seen as follows (see Figure 4.9.). Assume that all flightless birds (penguins, ostrich, kiwi etc.) account for 2 percent of

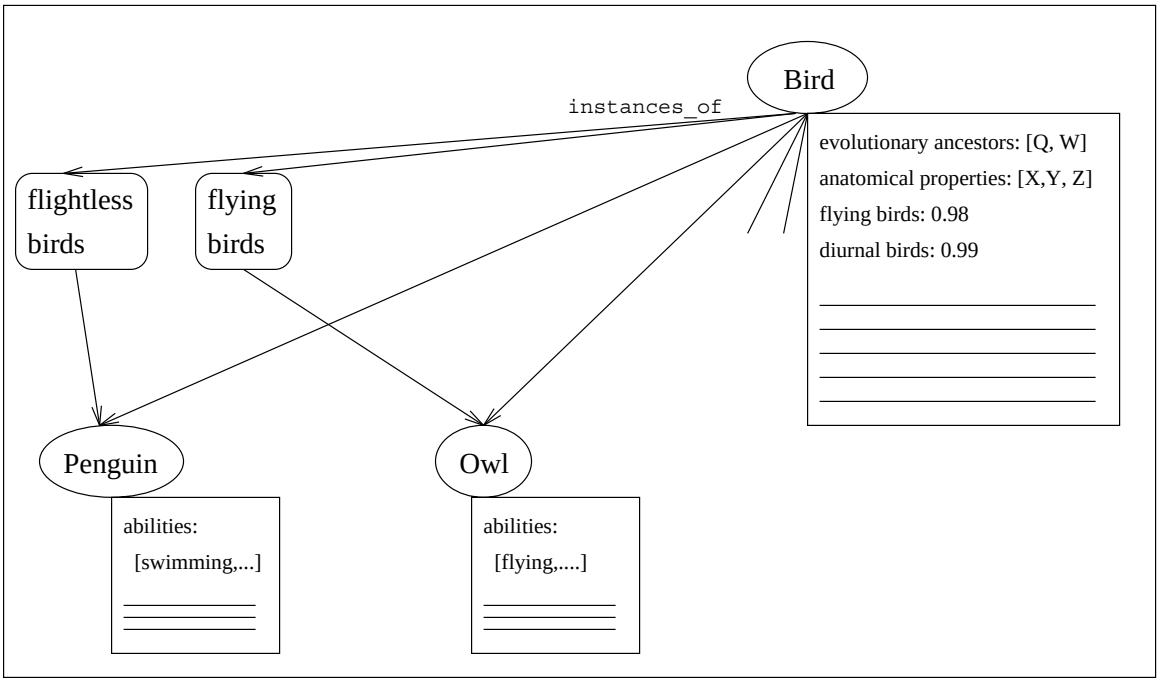


Figure 4.9. After the frequencies are introduced Ontological Network also allows reasoning mechanisms based on them.

all kinds of birds; i.e., 98 percent of birds fly. This can be represented by counting all instances of birds known in the Ontological Network. In other words, the qualities of a class are determined by its subclasses. We can call this *backwards quality propagation*,

or *retrospective inheritance*. In Reiter's "bird" example, if we know that birds fly and we also know that penguin is a bird and does not fly, then if we implement these both facts there is no easy way to reason whether Tweety the canary flies or not. It is obvious that if we don't know what kind of bird Tweety is we cannot reason whether it can fly or not.

Tweety can just as easily be an ostrich, for example, and saying that Tweety flies would be wrong in this case. "Canary" is also just a word, and computers do not know what it is unless we define it properly. Based on the common qualities of some birds, such as flying and not flying, we can build two classes and associate those birds correspondingly. So, we may say that there are birds which fly if they belong to that class of flying birds. If, however, we do not build those classes we still can say, based on the retrospective inheritance, that any given bird flies with 98% certainty; i.e., Tweety is in this 98%. Obviously, this needs reasoning mechanisms other than classical logic.[22] Probabilistic logic and fuzzy logic are two main topics among other knowledge representation areas that we need to further study for better Multifaceted Ontological Networks in the future.

[22]Reiter in his paper introduces the insufficiencies of classical logic in this case.

CHAPTER 5

FACETS FOR ONTOLOGICAL NETWORKS

Knowledge observed in the real world is analyzed through OSAS and represented in an Ontological Network. It is decomposed into numerous entities in various granularities, classes, and perspectives as well as in different degrees of precision. Since different individuals conceptualize those entities in different classes, the decompositions of knowledge contain many different perspectives at the same level of granularity.

Human perception and cognition are intricate processes. Human beings have survived in Nature, learned about their environment, and adapted to it very well through his accomplishments in those processes. Through OSAS, we attempt to represent the complexity of Nature in Ontological Networks on computers. Unfortunately, since our minds do not function in the same way computers do, it is impossible for any one of us to transfer an entire understanding of Nature to the computer. To compensate this we build facets through which he can represent his particular knowledge in Ontological Networks as well as retrieve knowledge represented on this platform.

In this chapter, we analyze the structure and functions of facets. In the first section, human cognition, which will interact with facets, is analyzed. The following

section presents the complexity of Ontological Networks to show the difficulty a user might have understanding it as a whole. In the next section, these analyses conclude in requirement synthesis to resolve problems at both human and computer sides for a smooth, productive, and discursive interaction; in short, the idea of facets. The remaining sections present techniques embodied by facets in order to realize the idea of facets and our requirements regarding efficiency in Human-Ontological Network interactions.

5.1. The Perception Problem of Conceptual Details

Human beings are able to filter out certain relevant stimuli from the millions of others that are irrelevant. Acquiring only the ones which originate from the entity being focused on is a skill that still amazes scientists. For example, when a girl enters in a room where she finds a table and an open book on the table, she immediately can distinguish the table from the room, and the table from the book on top of it. These, like many other observable entities in real world, vary in size, colors and other visual cues greatly; our ability to distinguish them has been developed throughout our evolution. These qualities along with others help us identify the entities around us. If that girl looks at the photograph depicting the same room, there would not be any difference in her perception.

However, if she has just some words on paper describing those entities, (such as "room, a table, an open book" and so), it would be a lot harder for her to comprehend

the situation in that representation since the words are not "visual" concepts. They are just symbols which are very similar in terms of granularity, shape, and color; thus, they are more difficult for a human being to perceive compared to natural things. Our senses have been tuned to decode stimuli coming from natural objects. We, nevertheless, push ourselves to decode other stimuli, understand unnatural things, and comprehending purely conceptual representations.

The same difficulty exists in human-computer interactions. When a computer outputs the entire spectrum of details about a domain, it is nearly impossible for us to distinguish the hidden features in which we are interested. Given such detailed description, *understanding* the represented world may not be possible for our practical purposes.

This is why abstraction is essential part of human thought. Since we cannot filter out unnecessary conceptual details the way we can filter out physical details, we must represent knowledge in a form that presents us with the only the most relevant facets from our point of view.

5.2. Solving the Perception Problem

The OSAS converts our informal knowledge to formal knowledge and represents it in an Ontological Network. Formal knowledge is precise and consistent but may be very hard to comprehend.

Various professionals investigate Nature in different granularities and details, in diverse, sometimes contradicting perspectives, in great variances of its conceptualization and evaluation, and in dissimilar formats, representations and terminologies. The artifacts and conceptual universe of one profession involve numerous implicit assumptions that have no coherence with those of other professions. Ontological Networks are designed to combine all those universes on the same platform, as Nature much does, by extracting *the most basic* building blocks of those universes and integrating them into a single complex conceptual universe.

An unfortunate consequence of this is that the resulting Ontological Network would be extremely complex. Distinguishing specific knowledge from such Ontological Networks could be intractable for our skills, which are tuned to the necessities of Nature. Therefore, we have to handle Ontological Networks in such a way that the information inside could be *pragmatically* accessible to users.

To do this, we need to conceptualize the world according to a few relevant criteria and limit the number of items under consideration to those criteria. However, we do not wish to limit the world to a single picture or model, as conventional representation techniques do. We need means through which we could see all aspects of the conceptual world rather than limiting it to a single abstracted model. Our windows to the OSAS world should always be flexibly changeable and be able to accommodate different *facets* of our world.

5.3. The Idea of Facets

Facets are our windows to conceptual universes in an Ontological Network. They are like fictious pairs of eyeglasses which are user specific, and through which users see whatever they need to see. Imagine the following scenario: There is a complex geographical map on which all geographical entities as well as socio-economic ones are illustrated; however, certain users are interested only in city names, while others might only be interested in lakes and rivers or railroad routes, or of socio-economic qualities of a population at a certain locations. Each user in this case needs a pair of such glasses which filter through only the subject of interest. Facets are very similar to those fictious glasses in this sense. If they are handled properly they show only the parts of interest of a highly complex Ontological Network (see Figure 5.1.).

The idea of a facet is not only necessary for acquiring knowledge from the Ontological Network representation, but also for representing knowledge within Ontological Networks. Since every professional has different view, each would like to state the world differently. Since the entire conceptual universe is composed of all those viewpoints, we should be able to acquire every perspective in order to complete our "jigsaw puzzle" to the fullest extent possible. Each professional would like to know how his piece overlaps those of others. When all put their pieces in, the superposed images would reveal *the entire world we know it.* In addition, the gaps on the picture will reveal specific areas where our knowledge is currently lacking.

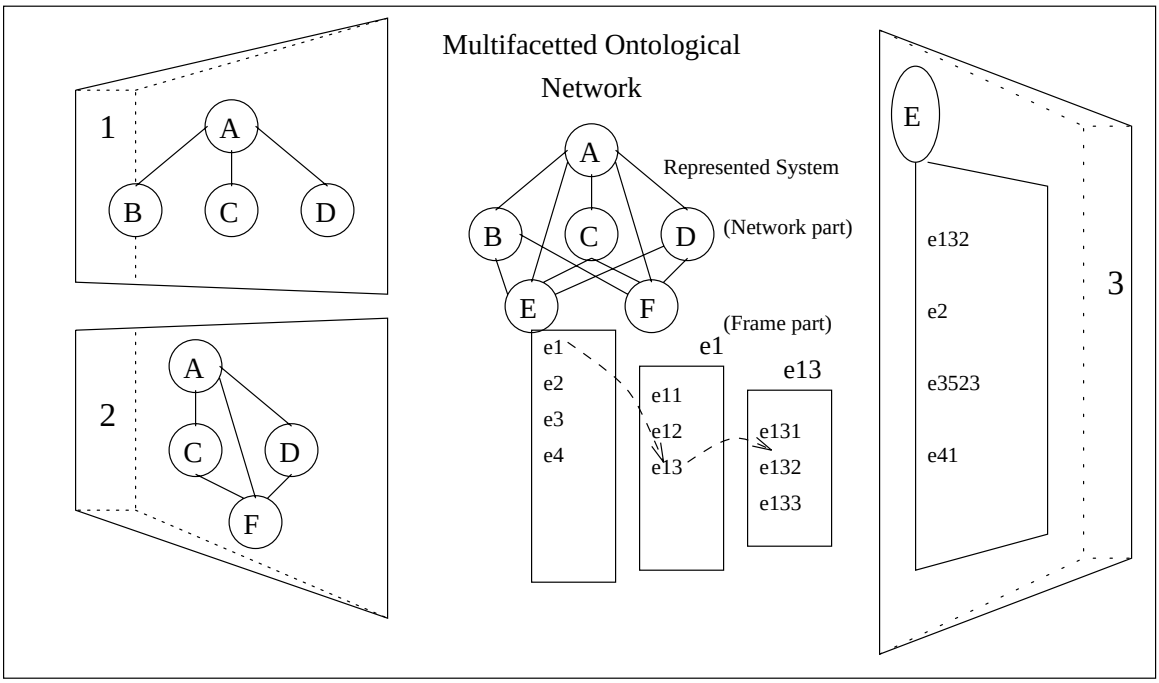


Figure 5.1. Viewing Ontological Network through different facets. Facet 1: A high level of *aggregation*. Facet 2: An *abstraction*. Facet 3: Retrieving only the desired levels of *precision* or desired formats of data representation.

5.3.1. Conflicting conceptual universes

In addition, scientists may not always be correct; sometimes they contradict each other, and it might not be easy to distinguish the right interpretation from the wrong one.[23] Inevitably, some of the represented knowledge will be inconsistent with others. This type of inconsistency is typical in high level organizations, such

[23]Sometimes however, both may be right, and the discordance may be just an illusion caused by the differences of conceptualizations. An analytical solution to this problem can be achieved by refining concepts or entities at lower aggregation levels of Ontological Networks.

as physiological or clinical ones, and prevent us from processing the knowledge as a whole. For acquiring knowledge in a specific context, every professional may use a certain *facet* that not only reflects his professional priorities and preferences, but also isolates any of his conflicting knowledge within his specific facet while it integrates the remaining non-conflicting parts of the different views.

5.4. Sharing Knowledge by Aggregation

The OSAS and Ontological Networks are not separate concepts. The analysis of the actual domain is accomplished by observing the relations of its entities and synthesizing (or reorganizing) through ontological perspective at the Ontological Network. The resulting representation is therefore considered as a synthesis based on the analyzed properties of entities.

Aggregation (synthesis) means incorporation of things and processes together by building a whole. The aggregate or the system is composed from two or more parts or components. Investigating a system by decomposing it into its components is called analysis. Finding the properties of those components and those of their interrelations and reaching a conclusive description based on that are processes that lead to synthesis. Those are the activities we do in Ontological Systems Analysis and Synthesis. Through Ontological Analysis, we search for *basic* relations. Ontological Synthesis corresponds to the aggregation of knowledge modules (system components) in the Ontological Network.

5.4.1. Holism versus Reductionism in the natural sciences

Although the characteristics of the whole is not the sum of its parts, their analysis gives us precious insight for the synthesis of the whole. This is a debate between two philosophical schools, holism and reductionism (Yates 1982). Physical and applied sciences always decompose systems of their domains, so that professionals can focus on individual problems. Many times, however, they neglect to reach a synthesis as this is the most difficult part of this whole process. On the other hand, the holistic approach claims the reductionist approach is misleading, and that one cannot understand the system without considering it as a whole.

Because of the methodological nature of pure sciences, many scientists may refuse the holistic approach. However, coming from the field of medicine, we think that it is not (yet) possible to synthesize the whole (of Nature) completely from its components. We are lacking in knowledge that is necessary to build the whole picture. Therefore, we need to analyze systems at different levels concurrently and try to establish the connections between systems and their components as much as possible, while representing each individual system as a whole as observed in Nature. At high level aggregation, we may relate some essential properties to the whole which were not observable in its parts. The differences between our synthetic system and the real world system will reveal our knowledge gaps, giving directions for further research. Investigating systems at different aggregation levels is not limited to the analysis but also to the representation and presentation of analytical knowledge within

the whole. Different aggregation levels in an Ontological Network will also help us to represent acquired knowledge at certain granularity where the relations of that knowledge with other levels cannot be seen clearly. So, the knowledge along with its yet unresolved problems can be preserved.

5.4.2. Aggregation of different views

In our representation we do not intend to represent one or few aspects of the domain, like a single discipline usually does, but all of them together. If a domain can be usefully decomposed in 100 different ways we would like to represent all 100 of them. In the Ontological Network, we represent *all* types of views by aggregating their components into a single system (see Figure 5.2.). This gives the most complete representation possible of a system, and differentiates the Multifaceted Ontological Network from other representations which do not incorporate detailed representations which are efficient to access from any point of view.

For example, if we ask different individuals how the entity "human" can be decomposed, each of them would probably do it differently (see Figure 5.3.). A person can identify the components of human being as head, torso, and limbs; yet another may conceptualize human as a collection of internal organs, muscles, bones, skin etc.; another person may see human an integration of physiological systems such as cardio-pulmoner system, digestive system, neural system etc. Nobody can say any of these views is wrong; all of them are correct. From a conservative point of view,

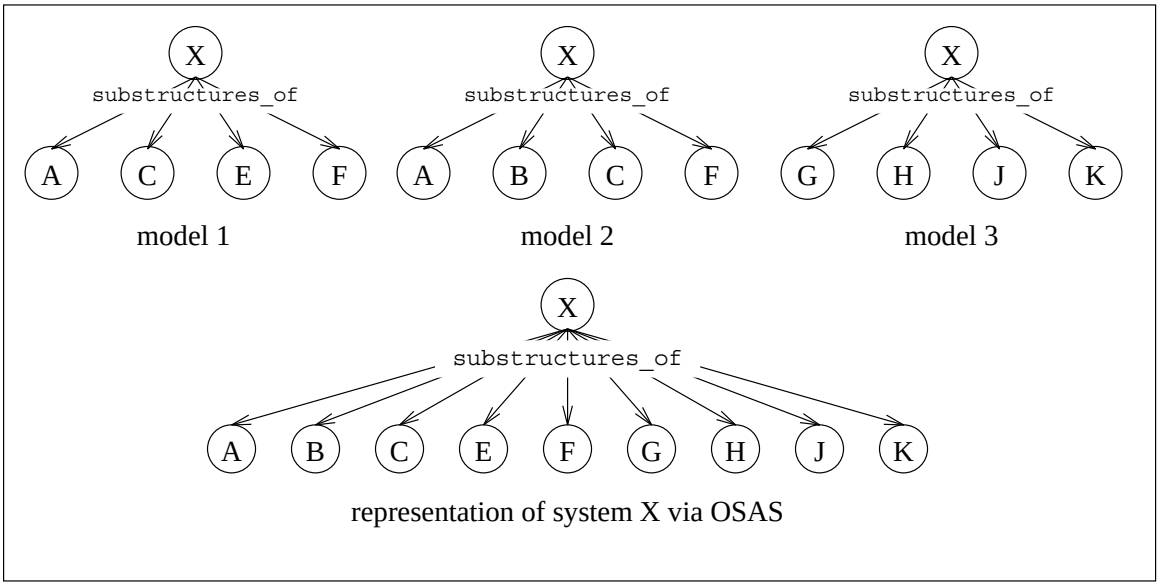


Figure 5.2. A system is an aggregation of all components decomposed (analyzed) by all (three) different views.

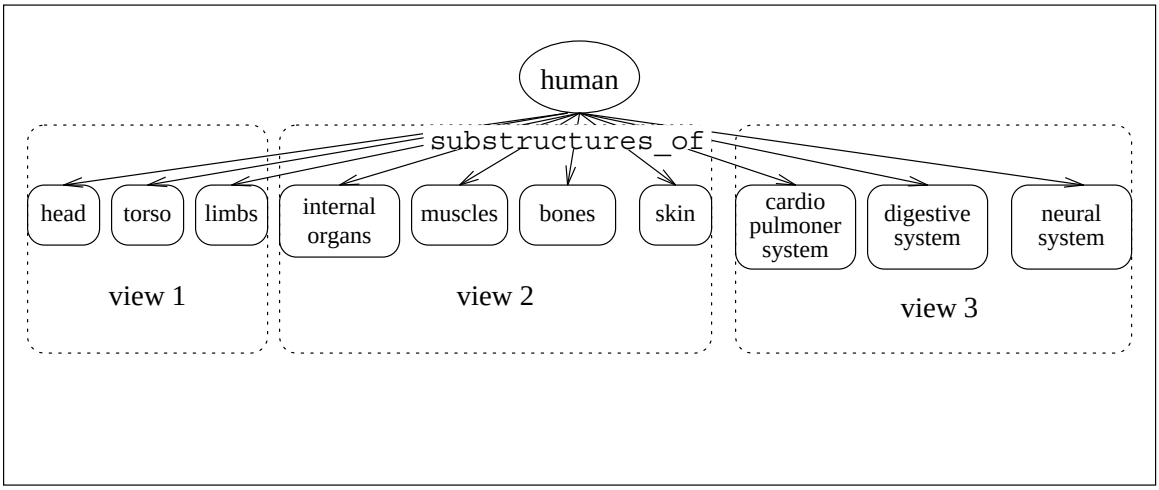


Figure 5.3. A bare representation of all three views on human being in the single representation system.

however, they are *incompatible*; therefore, we need to choose one of them, whichever one is most suitable for our *specific* job, and stick with it. From the OSAS point of view, they are compatible in the real world, so they will be in our representation system as well.

5.4.3. Reconciliation of different decompositions

The first action to take is to create the representation of the domain, because we cannot relate these seemingly incompatible views before we represent them all together on the same platform. Given that underlying platform even at just a few levels of detail, one could see how this system would be compatible (see Figure 5.4.).

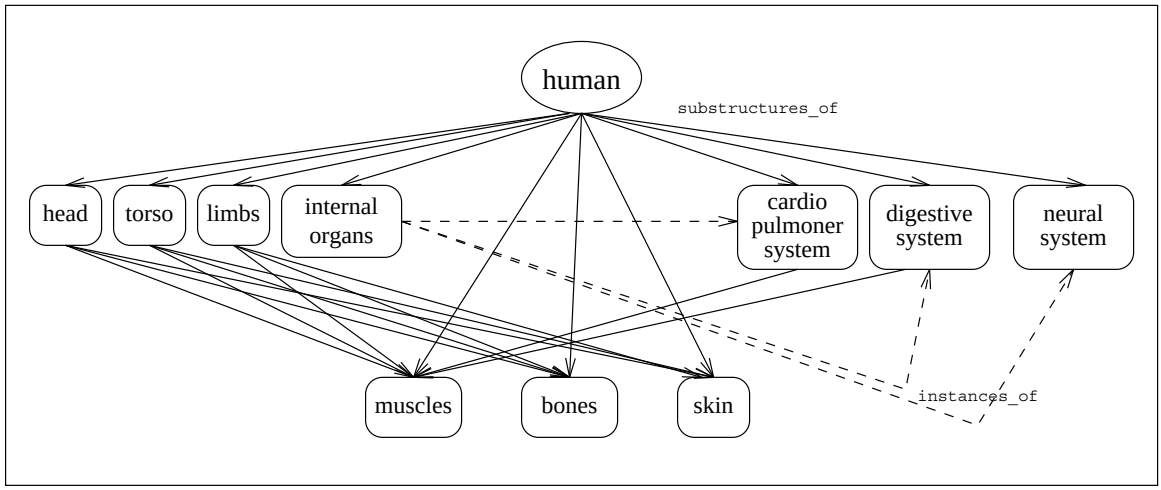


Figure 5.4. Establishing relations between different types of systems decompositions. Continuous lines are `substructures_of` relations whereas the others are `instances_of` relations.

The head, torso, limb type decomposition is a topological view and would help us to locate the subsystems throughout the whole organism. The physiological systems will help us to see the functional relations between the organs as time passes. If we further decompose these physiological system-type decompositions, we could identify those organs already embedded there. (see Figure 5.5.).

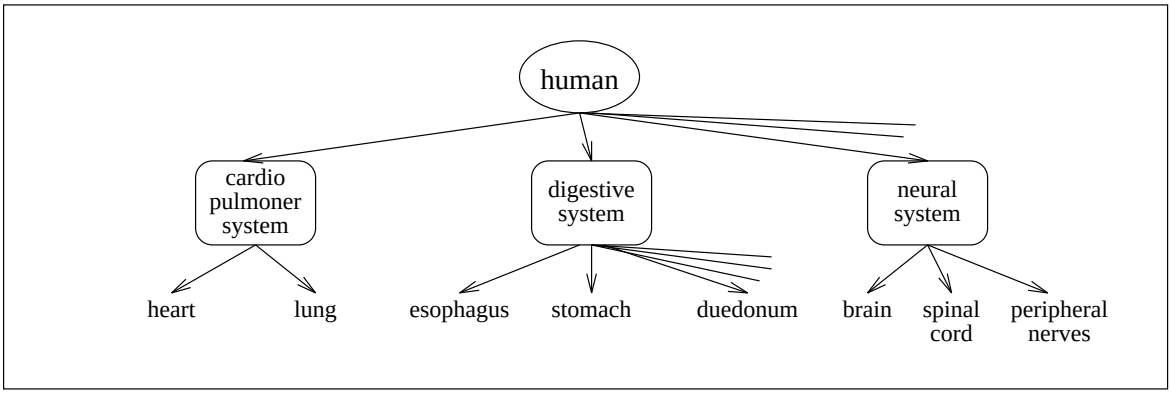


Figure 5.5. Physiological system type decomposition merges with organ type decomposition in few steps.

In further detail, it is seen that every type of decomposition uses the same building blocks; that is, the *basic* reusable knowledge modules are *shared*. There is a great deal of shared knowledge between different types of decompositions of subgraphs (see Figure 5.6.).

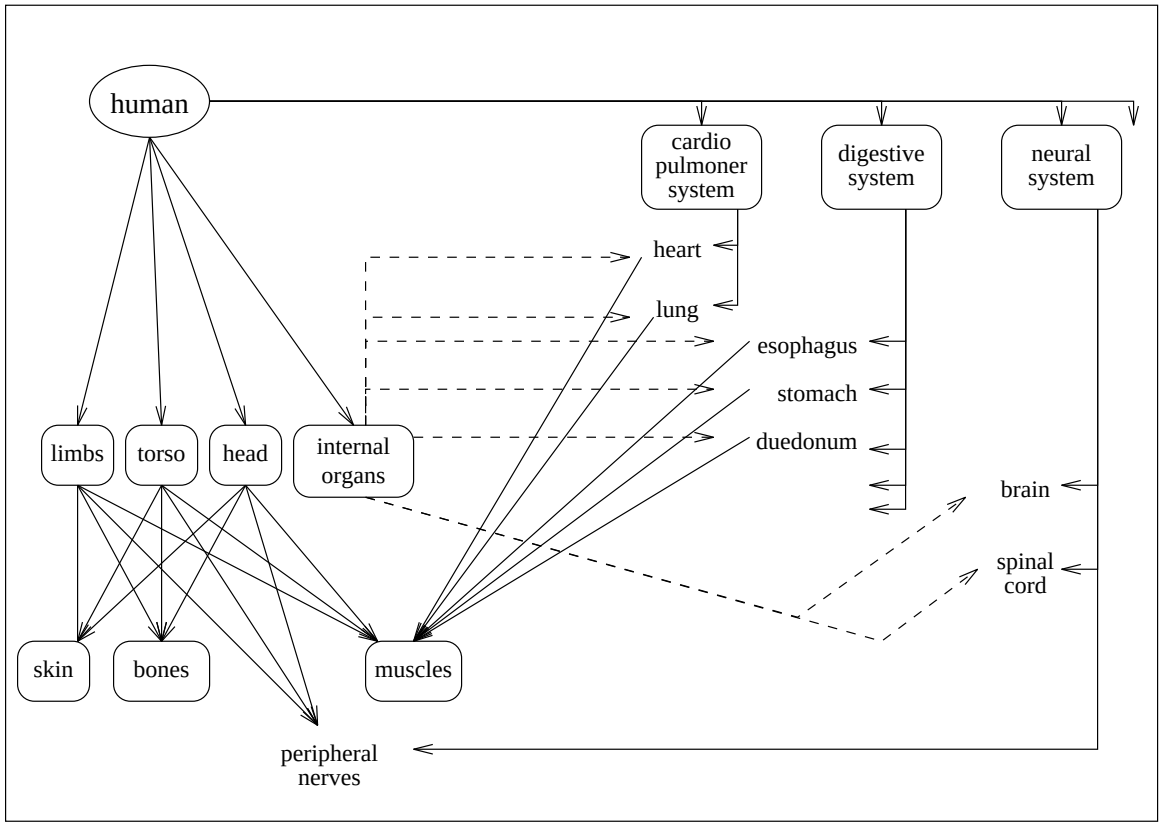


Figure 5.6. Different decomposition types merge to the same building blocks, shareable and reusable knowledge modules. Straight lines are `substructures_of`, dotted lines are `instances_of` relations.

5.4.4. Presenting knowledge at different levels of detail

A useful property of facets is determining the level of detail or the scope of the user. For example, the scopes of the internist and of the researcher in a laboratory are certainly different. An internist would not be interested in minute details of a biochemical reaction and its assay methodologies, while the clinical scope of a researcher would be narrower and in greater depth than that of the internist. This

detail can be limited to a certain desired level by not allowing the user to view an entity beyond that range of granularity.

Everybody has their own level of interest in the same domain. A biochemist would not like to see a very high level knowledge such as "biochemical processes occur in living organisms." This is irrelevantly high level for him, as he is in need of very specific detail. On the other hand, a high school student may be interested in that high level knowledge but not in the lower level details.

For example, assume that we have the following process decomposition tree (see Figure 5.7.). If A41 is searched by a user, the system would generate a path in a list

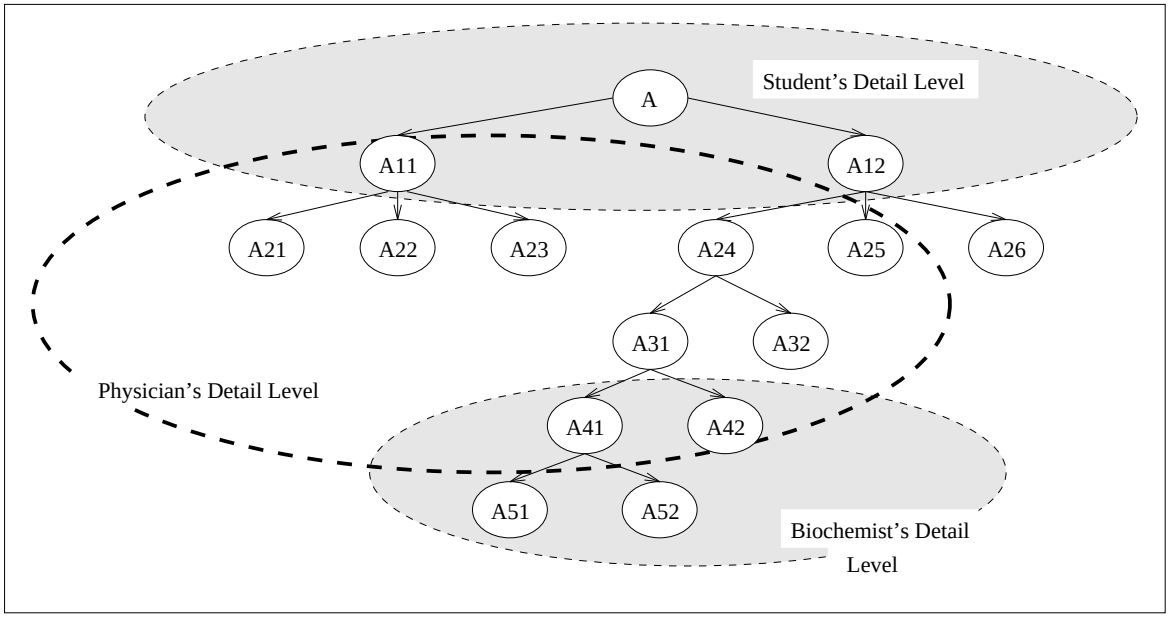


Figure 5.7. Different individuals may need knowledge in different levels of aggregation (detail).

format from A41 to the root of the decomposition tree (which is the node physical process).

```
[A41,A32,A24,A12,A, ... ,physical_process]
```

If the user is a student, then the answer would be "A41 is a detailed subprocess of A12;" however, if it is a medical doctor, then the answer is "A41 is a subprocess of A32." In other words, A41 is invisible to both users since it is too detailed for them; therefore, they both have received the processes at the limits of their detail levels. If, however, a biochemist asks the same question, he would retrieve the necessary subgraph consisting of A51 and A52.

5.5. Sharing Knowledge by Abstraction

Focusing attention on certain parts of a domain while neglecting its other parts is called abstraction. Through observations, human being acquire knowledge from the environment. That knowledge is also called a Mental Model, which is *an abstraction* of the real world (see Figure 5.8.). Abstraction is a simplification method used in every activity of human beings, such as mathematical modeling (Carson, Cobelli, and Finkelstein 1983). Abstraction in mathematical modeling frees us from many unnecessary details and enables us to estimate the nature of the modeled entity.

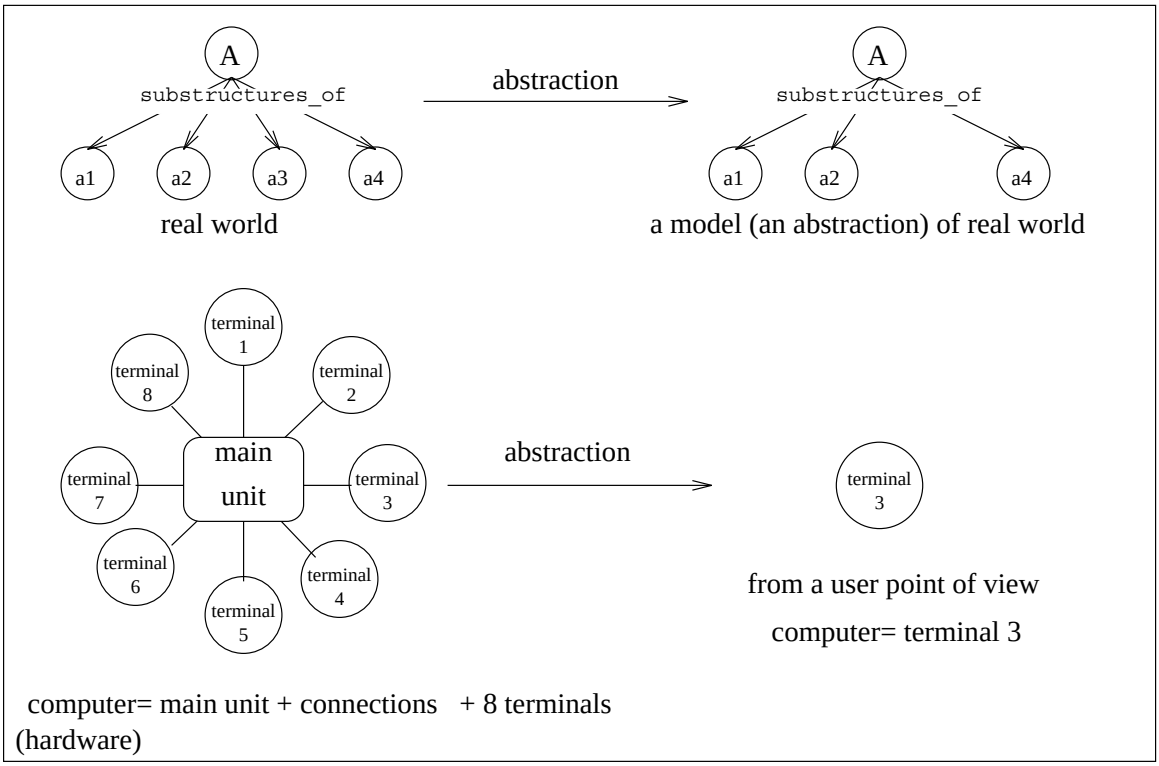


Figure 5.8. Abstraction is conceptualization of an entity with only its *certain* parts or qualities.

5.5.1. Information hiding

A common usage of abstraction in databases is information hiding. For example, there may be many items stored in database of companies. A particular user, however, may only be able to reach certain types of items. It is not desired, for instance, to allow the cashier to see the manager's salary. Every user has a certain abstraction of the database, but the actual system is aggregation of all components seen by different users.

In Ontological Networks, the same technique is used extensively in facets. Viewing the intended relation is specified in facets. If there are relations which are extensively utilized by a certain class of professionals, then by labeling those relations and attaching their labels to the facet records of professionals may we maintain the extents of facets. If three professionals decompose the entity "A" into three different abstractions, then there would be three facets (see Figure 5.9.). In this example, we

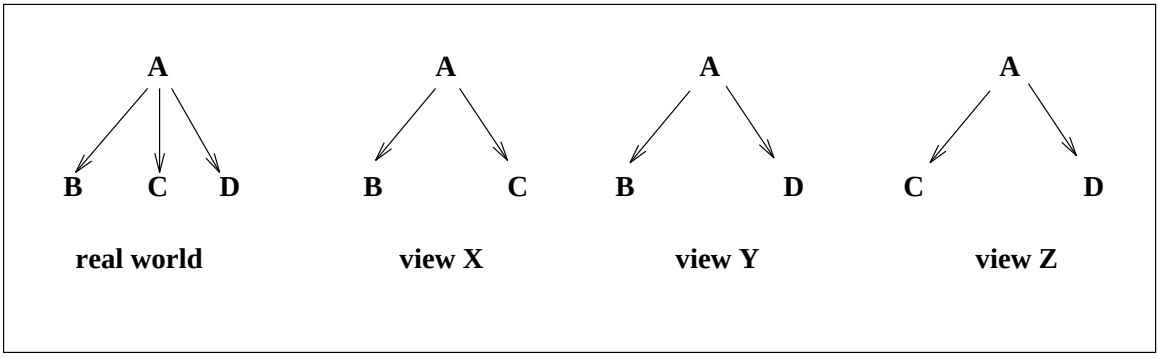


Figure 5.9. Three ways of decomposing "A" through abstractions.

can label the relations with facet numbers in the form of:

```
facet(<predicate>, <facet no>, <list of facet users>)

facet(relation(A,B),f1, [X,Y])

facet(relation(A,C),f2, [X,Z])

facet(relation(A,D),f3, [Y,Z])
```

Each view, in this specific case, uses two of these facets. Those facets can be represented in a profession-specific facet frame with facet numbers (see Figure 5.10.).

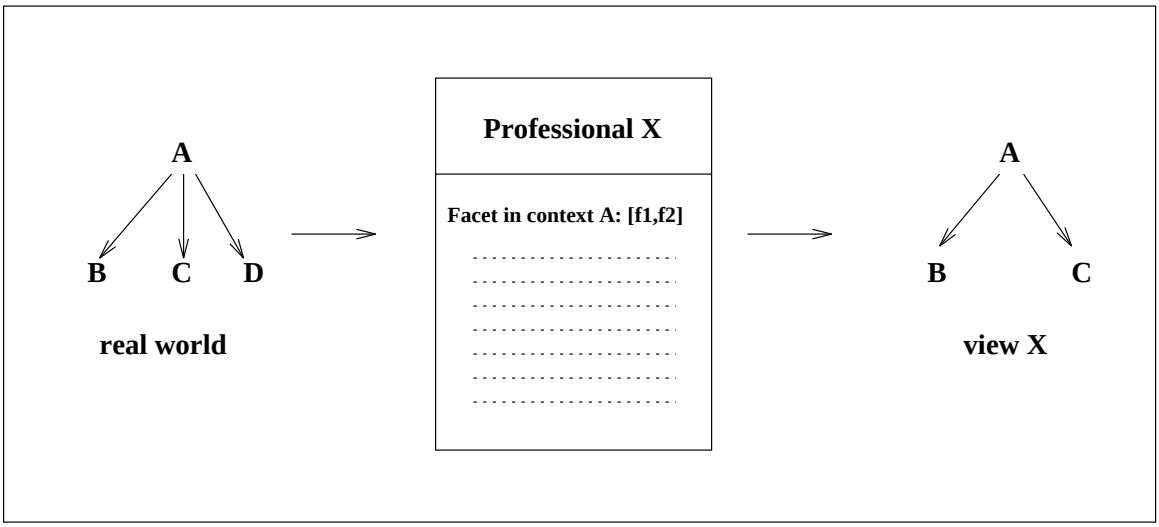


Figure 5.10. Each professional has a frame in which the facets he uses are recorded.

The relevant slot may be indexed by context name or by highest level node in the subgraph embodied in that facet. Whenever that part in the network is visited (or questioned), the frame slot should be activated and the list of the facet numbers at that slot should be used to view the desired part of the network. The facet frames, in this case, act like a hash table data structure.

5.5.2. Reaching other user's abstractions

However, this alone would not be sufficient to utilize Ontological Networks. A user may not be satisfied with the view of his facet and would like to see the facets of other professionals in order to get feedback. In addition, if he has not established his

facets yet, he should view first the existing facets. If one of those already represents his view perfectly he may just reuse them. This can be executed as follows: If the user is interested in the entity A (see Figure 5.10.) and considers that B is an essential part of it, then he needs to see facets which at least include `relation(A,B)`. In the above case there is a single facet with that specification:

```
facet(relation(A,B), f1, [X,Y])
```

There are two users of this facet. In order to see the views of those users, their facet frames and then their facets should be retrieved. By tracing through the specifications (labeled predicates) in those frames, the user can consult the views of professionals X and Y.

5.6. Sharing Knowledge by Idealization

Two different professionals, e.g., a physician and a biochemist, might be satisfied with different levels of precision in the data they acquire. The internist is interested in the variation of the data between physiological and pathological values and whether it has increasing or decreasing tendencies under the chosen therapy schedule. For him, data which presents these criteria sufficiently is precise enough; on the other hand, the biochemist may be interested in any minute value fluctuation of the same clinical data. This idealization also extends to the number of qualities as well as the precision. A physician might want to reach certain biostatistical data of a particular population of which specifications match the ones of his patient. This process is conducted by

user facets via their labels indicating whether that specific data is within their scope or not.

Perhaps the weather example illustrates idealization by facets better than any other example. Let X be a traveler flying from Dallas to Quebec who is interested in the weather forecast. He expects to hear whether the weather would be sunny or rainy in Quebec. On the other hand, let Y be a researcher in Cape Canaveral air base who also is interested in the weather forecast. He, however, needs to get data regarding the speed of the wind along with its directions in 15 minutes intervals (see Figure 5.11.). Both access the same entity in the network, and activate the same knowledge process but X gets only the high level knowledge, while the Y gets detailed knowledge with many parameters.

5.7. Sharing Knowledge through Terminological Mapping

Through the specifications within the facets, a user may use all of his conventions within the system, including terminological conventions. By keeping a thesaurus which includes synonyms of the vocabulary used originally in the system, the user's terms would be interactively modified to translate back and forth from the system's terminology to the user's terminology (see Figure 5.12.). Input coming from system activates the frame called User Lexicon. If the word is not in Lexicons of the user (User's specific lexicon + other lexicons that user utilizes), it is checked in thesaurus for the synonym in user's terminology for substitution in user's view. If user inputs

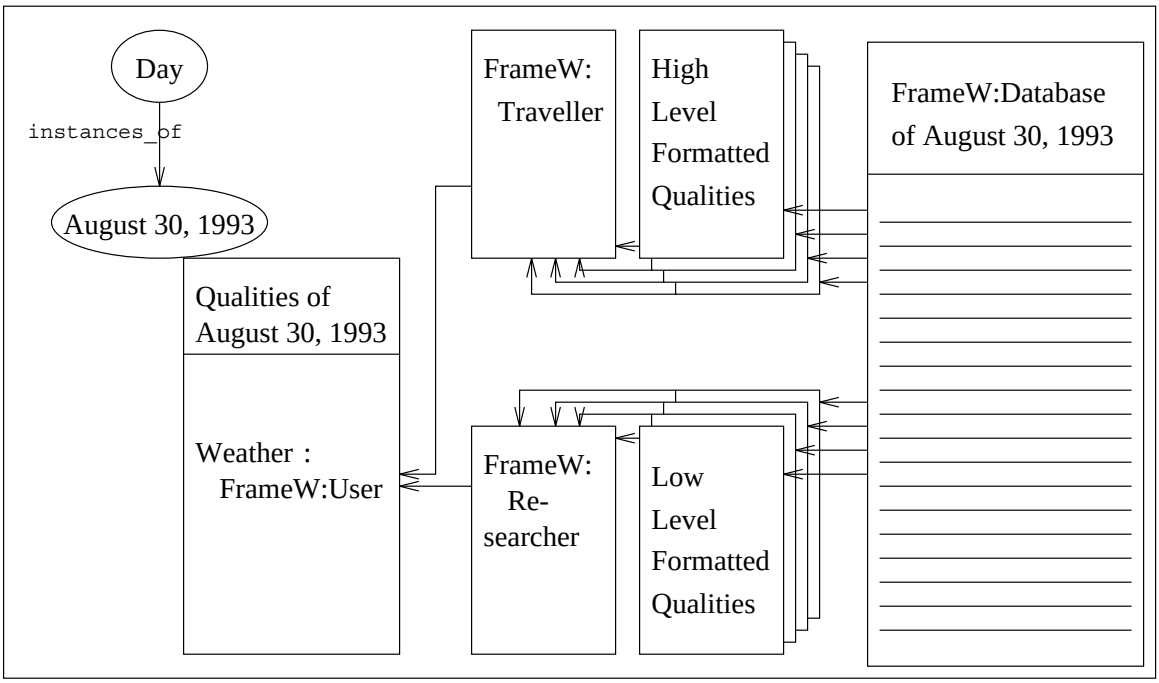


Figure 5.11. Researcher's need is combined through low level data which themselves are combined from the database, while the traveler's knowledge need is acquired through processing data in high level formats.

a word that is not in Common Lexicon then it is checked for all thesauruses for the synonymous word usable in system.

In our implementation, we kept a single thesaurus for everybody. It is in a simple list format, where the head of the list (`nicotinic_acid` in the example below) represents the word used within the original system.

```
synonymous([nicotinic_acid, nicotinate, vitamin_B3]).
```

In this way the user may ask questions or provide information through synonyms

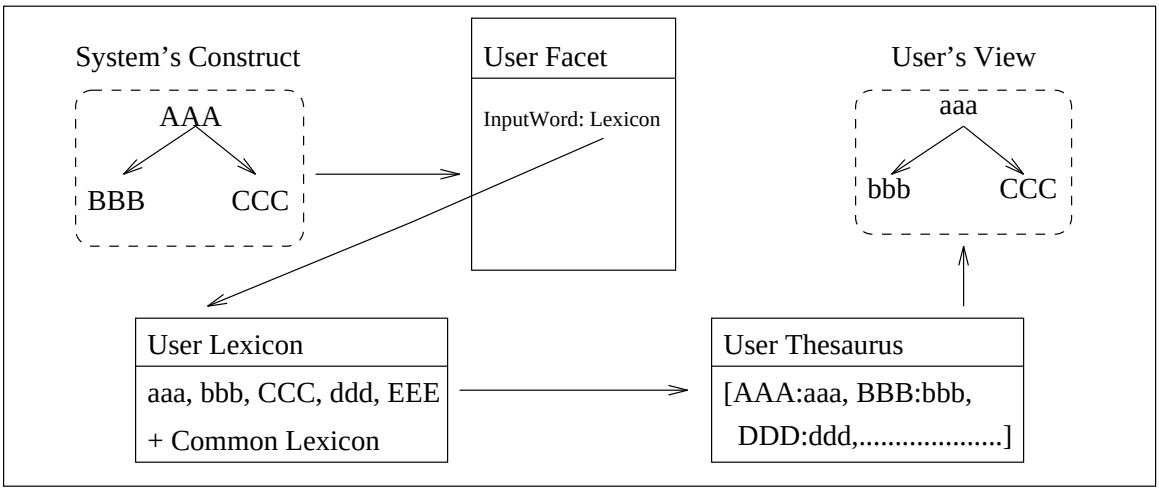


Figure 5.12. Terminology mapping.

of existing words. In facet implementations, every term is used uniquely in certain facets and enumerated accordingly, such as

```
synonymous(
  [nicotinic_acid,
   nicotinate:facet12:facet3c,
   vitamin_B3:facetA2]).
```

Using this thesaurus, a user of facet12 or facet3c would see the term `nicotinate` instead of `nicotinic_acid` that is the actual term used in the system. The same would happen if the user represents knowledge on this platform, namely his term `nicotinate` would be translated to the `nicotinic_acid` before it is placed.

5.8. Sharing Knowledge by Integrating Inconsistent Views

There may always be some users with controversial views. Their facets would be invisible to other users and the commonly agreed relations in certain contexts would be invisible to those people unless they wish to view those relations (see Figure 5.13.). In this example, the entities are related to each other with two kinds of relations, rel1

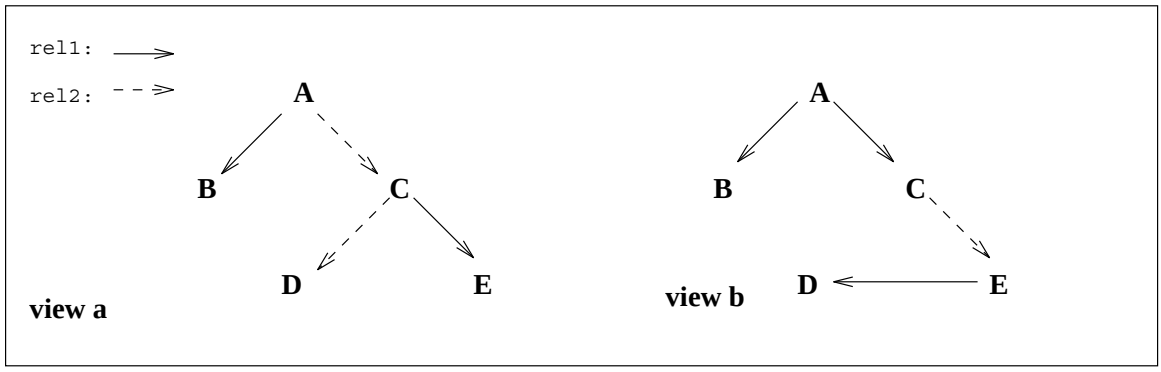


Figure 5.13. Two different conceptualization of relations between entities A, B, C, D, and E.

and rel2 that are illustrated with different line styles in Figure 5.13. Obviously, the two views in this example are not compatible with each other. One way of implementing these views at a logical level is as follows.

```
                facet(rel1(A,B),[1a,2b]),
%% 1a and 2b are user identities
facet(rel2(A,C),[1a]),      facet(rel1(A,C),[2b]),
facet(rel1(C,E),[1a]),      facet(rel2(C,E),[2b]),
```

```
facet(rel2(C,D),[1a]),      facet(rel1(E,D),[2b])
```

As is seen in Figure 5.13., the picture is easy to understand; the actual network would look different without facets and might be much more difficult to acquire knowledge from (see Figure 5.14.).

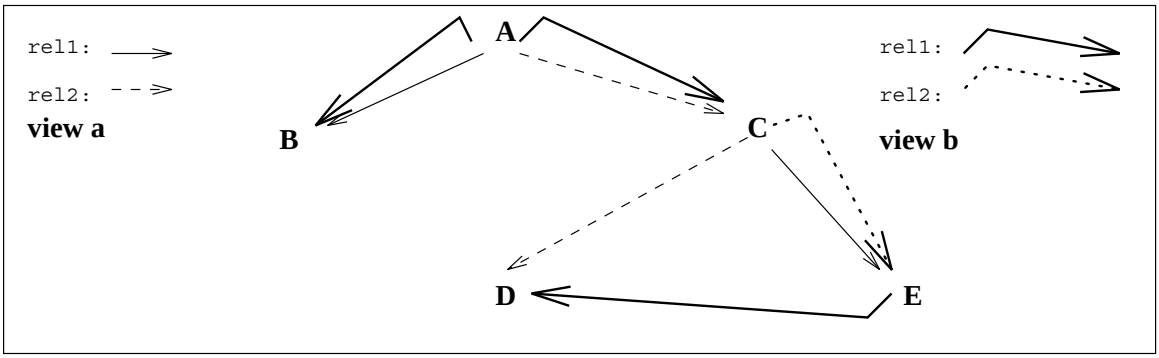


Figure 5.14. Facets enable us to see the patterns of relations. Without facets, acquiring knowledge from the representation may be very difficult.

CHAPTER 6

CONCLUSIONS

6.1. An Answer to Model Selection Problem

There is a problem in model-based reasoning, called *model selection*. Randall Davis classify this as one of the four open issues in this area:

> *"Given a basic knowledge of how to use models of structure and behavior in diagnosis, it is intriguing to push the process back one step: How are models selected to begin with? Since all devices can be viewed from multiple perspectives, how do we decide which view is appropriate in any given circumstance? The is the sort of reasoning that goes on in the heads of engineers before the equations or block diagrams ever hit the page. Interesting starts on this problem have been made (e.g., (Addanki, Cremonini, and Penberthy 1991; Falkenhainer and Forbus 1991; Weld 1989), and others), but much remains to be done."* (Davis 1993)

Multifaceted Ontological Networks may be an answer to this problem, because the models are totally integrated instead of separately built. Therefore, the problem is not selecting a model (i.e., context changing) any more, but instead selection of the appropriate facet so that necessary components can be reached. Determining the appropriate facet in this context is not the same problem as in model-selection, as the most appropriate facet is the facet whose specifications match the specifications of

the problem as a whole. The problem specification may consists of certain relations and/or certain entity names. The matching process may be accomplished by searching for these within the network. The appropriate facets will be the one(s) which includes those entities and relations at that aggregation levels.

Since individual models are isolated views of the world (see models on Page 93 and 104), the most appropriate model is selected based on the requirements of the problem. Because there is no meta-model among them, the selection is limited to the conceptualization of a knowledge engineer working on problem. Since the models in this context are created from different views and design commitments, they may not share the same specification language if a formal one is used.

In Ontological Networks the models are not separated, therefore there is no proper model selection problem. The requirements of the problem (the variables and parameters of the problem) should be unified with the specifications (relations, entities and their qualities) of the knowledge modules in Multifaceted Ontological Networks.

6.2. Summary

The amount of information available to us has increased exponentially and will always continue to do so. Since our information processing capability is highly limited, we cannot fully use the available information and extract hidden knowledge from it. In order to fully utilize our information sources we need to rigorously define

represented knowledge for our computers. The only way that has been found so far involves decomposing the real world entities into its parts and subclasses, and establishing the relationships between them completely and formally. This activity is called (formal) Ontological Analysis.

Ontological Analysis enables us to define the entities rigorously, since every entity is explicitly built up on top of other subentities. Therefore, there is no place for ambiguity in interpretation as long as those subentities themselves are defined rigorously as well. This rigor let us express and process complex concepts in computers. Since we can build a model of the world as we know it in computers in this way, the rules and procedures implemented in different software systems can be totally integrated with that model as ontological facets. This will broaden the narrow spectrum of intelligent computing that we see currently in conventional expert systems.

In our ontology-based knowledge representation, we establish only the essential formal relations between entities. Therefore, there are no artificial design commitments established due to practical reasons. The goal is to capture the world of entities as simply and purely as possible. The only commitment is labeling entities with certain English words, which can easily be translated into any other language or terminological conventions via a thesaurus or lexicon system.

No knowledge representation system based on a single view can avoid certain design decisions (ontological commitments) based on a particular world view and classification convention. Because of the introduced multifacet systems, we do not

need to commit to that type of design decision. We can capture every expressed world view within our representation system via user specific facets without contaminating other world views, while utilizing established relations by those views in their entirety.

Various levels of precision in data representations and their multifaceted management prevent us from having to commit to a certain format of data representation. This ability, along with the rigorousness in the ontological entity representation, enables our system to share knowledge with other software systems attached "symbiotically." We can assume this flexibility of interaction with other software systems because every professional (each with their own world view) can interact with this system.

In a categorical view towards the organization of our ecosystem, we see that the lower organizations determine the operation of higher organizations; thus, whenever we analyze a system we need to base it on certain other lower level systems. Therefore, it seems to us logically to start from very lower levels and reach a certain degree of completeness of those levels. That would give us feedback concerning the effectiveness of our representation. Based on this view, we have started to analyze biochemical and micro-anatomical worlds such as Citric Acid Cycle in mitochondria. In lower level organizations such as biochemistry, the certainty of our knowledge is higher and easier to quantify. Thus it should be easier to observe the deficiencies of our methodology and to refine it at that level.

This process of making the represented world more complete has no end. Therefore, the analysis and the synthesis should be performed hand in hand. Our entire enterprise in natural sciences is to search for the hidden relations between entities. In conventional scientific representation methods, recognizing any unknown relation is not an easy process to accomplish. This representation system enables us to analyze the observable relations as well as to make it explicit that certain relations cannot be established because of a lack of knowledge.

6.3. Future Work

Our methodology can also be enhanced to synthesize new relations based on the implicit (hidden) knowledge represented in the system. For this purpose we need to utilize the tools of classical logic. Since the system is presumed to be designed in a consistent and monotonic fashion, utilization of classical logic should not be a problem. In addition, its philosophical and logical background is well studied under the title of Formal Ontology (Smith and Mulligan 1982; Cocchiarella 1991).

6.3.1. Time and energy

Time and energy are two concepts we have not modeled as detailed as structures, compartments and processes, though time is indirectly represented in processes. In the current state of our Multifaceted Ontological Networks, structure is a being or a state of that being frozen in time; a process on the other hand puts structure(s) in

a time frame without explicitly mentioning the time span necessary. When we use biophysical submodels in our Multifaceted Ontological Network we must then deal with time explicitly so that we can make use of those differential models.

If a compartment is part of a process then we should also represent any changes of the three dimensional shape of the compartment. The spatial relationships should be improved in order to represent such shapes. Geometrical modeling expertise should be considered at this problem.

In this work, energy has been considered much less than time. We have used it only as an amount of supplied (or freed) fuel in a process. Expertise in physics is needed to represent energy as it needs to be.

6.3.2. Nonmonotonic and probabilistic reasoning

Scientific ontology is based on the scientific rules that are extracted by observing Nature. It is an outcome of certain methodological processes; therefore, different scientific methodologies may sometimes not result in compatible outcomes and ontological relations. In a methodological and theoretical approach, observations must be compatible. If a rule is broken once, then it is not a scientific rule anymore. In short, there are no *exceptions* in scientific studies. Therefore, we do not think that we need to base our Multifaceted Ontological Networks on exception-based nonmonotonic relations, and we do not establish the concept of an "exception." However, we know that things in Nature are not always black or white, or deterministic.

There is a causality in the chain of biochemical reactions. It is based on certain conditions,[24] and the probability of coming across of two molecules in a certain compartment where the biochemical reaction occurs. Adding these probabilities to our representation would help us to solve the qualitative and frame problems (Hayes and Ford 1992). We need to investigate fuzziness of qualities and probabilistic process representation.

6.3.3. Methodology for quality representation

The types of quality representation should be investigated further. We could not see any quality pattern that is generally shared, such as "everything has a color." A systematic representation methodology for qualities would be very difficult, if not impossible to create.

Representing qualities at different levels of fuzziness is another issue to be studied. This requires format conversion functions that may vary from a single table to a complex statistics software or a neural network system for clustering and pattern recognition.

[24]Conditions can be represented in the last argument of the predicate `process` of Multifaceted Ontological Networks.

6.3.4. Integration of different reasoning systems

Consistency or truth maintenance within the representation is a function of a reasoning mechanism based on certain metarules and premises. Those rules may be stated explicitly and the underlying representation can be tested through that. However, the question we need to consider is "Are those rules absolute?" Actually, in our TCA cycle implementation we also followed some metarules. One of them is: "If X is substructure of (or subprocess of) Y, then there must be no Y that is substructure of (or subprocess of) X." We think this is a clearly universal rule. Therefore, in our directed graph we always traversed from the root (e.g. ecosystem) down to the leaves for each path and made sure that it has been partially ordered (i.e., there is neither a cycle nor a redundancy on any single path). Other consistency checks can be subjective to the user. In Multifaceted Ontological Networks, different user facets do not have to be consistent to each other, but if there is a consistency rule which no one disagree with and if a test of system consistency through that rule reveals an inconsistency, then that erroneous entry should be discarded. The problem to be studied in the future is what general rules (such as the one we stated above) should be used to test our knowledge system.

Several different reasoning and truth maintenance systems must be connected to the first reasonably mature prototype of Multifaceted Ontological Networks so that the sharability claims presented in this work can be ostensively proved.

The implementation of the whole system should be extended in depth and in breadth. The implementation of facets has been started but not completed. To implement facets extensively, a domain (other than TCA cycle) that involves many different views may be used.

APPENDIX

We here would like to show the current state of our implementation. Since we did not complete our specifications on facets yet and the relating code used for test purposes cannot be integrated with the following one, we did not include it here.

```
:-[mylib, syn, isa, structure, instance, process, compartment, neighbor].
:-dynamic user_facts/1.

medKAT:-
  cls, for(12,nl),
  repeat,
    initialize_new,
    start,
    move_file('new.pl','session.pl'),
    !, restart.

start:-
  ask_word('Please enter the NAME of the ENTITY',Entity),
  ( check_input(Entity,[])
  ; indexing(Entity,_)
  ),!.

indexing(ecosystem,structure):- continue.
indexing(physical_process,process):- continue.
indexing(Entity,Type):-
  ( is_there(Entity,Syn,Type),%% Entity has been entered into kb before
    disp_isa1(Entity,Syn,Type),
    ( var(Syn),!,
      ( \+ rootchk(Entity,Type,_),!,
        networking(Entity,Type)
      )
    ; nonvar(Syn),
```

```
      ( \+ rootchk(Syn,Type,_),!,
        networking(Syn,Type)
      )
    ),!
  ; cls, %% Entity was NOT entered before
    get_type(Entity,Type),
      %% writing into the session's current file "new.pl"
    writef('new.pl', isa(Entity, Type)), !,
      %% A process should be specified with its components
    ( Type == process,!,
      add_process(Entity)
    ; true
    ),
    networking(Entity,Type)
  ).

indexlist([],_Type).
indexlist(List,Type):-
  member(S,List),
    indexing(S,Type),
    fail
  ; true.

networking(Entity,Type):- ( Type == structure; Type == process),
  repeat,
    ( set_in(Entity,Type)     %% Superclasses
    ; rooting(Entity,Type)    %% Superstructure(Superprocesses)
    ),
    ( path(Entity,Type,Path),     %% Path between Entity and Root
      message14(Entity,Type,[Entity|Path])
    ; message15(Entity,Type,[Entity|Path])
    ),
  !.

add_neighbors(Entity):-
  disp_isa2(Entity,structure),
  yesno('Would you like to enter its neighbor structures',no),
  write('Adjacent (neighbor) structures to '),writedq(Entity),write(' are,'),nl,
  ask_list(AdjacencyList:_,1),
  writef('new.pl', neighbor_structures(Entity,AdjacencyList)),
  indexlist(AdjacencyList,structure).
```

```
add_process(Process):-
  disp_isa2(Process,process), message6(Process), ask_list(Ls:_, 1),
  disp_isa2(Process,process), message7(Process), ask_list(Lp:_, 1),
  disp_isa2(Process,process), message8(Process), ask_list(Le:_, 1),
  disp_isa2(Process,process), message9(Process), ask_list(Lc:_, 1),
  writef('new.pl', process(Process,Ls,Lp,Le,Lc)),
  indexlist(Ls,structure),
  indexlist(Lp,structure),
  indexlist(Le,structure),
  indexlist(Lc,_),
  ( message2(Process),!,
    where(Process,SpaceL),
    indexlist(SpaceL,_)
  ; true).

where(Process,SpaceL):-
  disp_isa2(Process,process),
  write(Process),
  write(' occurs in following Compartments and Structures:'),nl,
  ask_list(SpaceL:_,1).

rooting(Entity,Type):-
  message10(Entity,Type,SuperL),
  member(Super,SuperL),
    ( Type == structure,
      writef('new.pl',substructures_of(Super,[Entity]))
    ; Type == process,
      writef('new.pl',subprocesses_of(Super,[Entity]))
    ),
    fail
  ; true,
  indexlist(SuperL,Type).

set_in(Entity,Type):- message11(Entity,Type),
  ask_superclasses(Entity,SuperClassL:_,1),
  SuperClassL \== [],
  indexlist(SuperClassL,Type).

decomposing(Entity,Type):-
  message12(Entity,Type),
    ask_list(ListSub:_,1),
    check_list_subparts(ListSub,SubParts,Entity),
    ListSub \== [],
```

```
    ( Type == structure,
      writef('new.pl',substructures_of(Entity,SubParts))
    ; writef('new.pl',subprocesses_of(Entity,SubParts))
    ),!,
    indexlist(SubParts,Type)
  ; true.

set_of(Entity,Type):-
  message13(Entity,Type),
    ask_subclasses(Entity,ListSub:_,1),
    check_list_subclasses(ListSub,SubClasses,Entity),
    writef('new.pl', instances_of(Entity:all,SubClasses)),
    indexlist(SubClasses,Type)
  ; true.


path(Entity,Type, Path):-
  consult('new.pl'),
  rootchk(Entity,Type,Path).

rootchk(ecosystem,structure,[]).
rootchk(physical_process,process,[]).
rootchk(S,structure,Pt1):-
  find_superstructure(S,So),
  Pt1 = [So|Pt2],
  rootchk(So,structure,Pt2).
rootchk(P,process,Pt1):-
  find_superprocess(P,Po),
  Pt1 = [Po|Pt2],
  rootchk(Po,process,Pt2).
rootchk(S,Type,Pt1):-
  find_superclass(S,So),
  Pt1 = [So|Pt2],
  rootchk(So,Type,Pt2).

update(Entity):-
  cls,
  write('Please be aware that you cannot update entries of this '),
  write('session but all earlier ones.'), nl,
  write('If you wish update any information entered during this '),
  write('session,please first'), nl,
  write('finish this session and update it afterwards.'),nl,
    %% Each entry is considered "independently" because we may deal with
```

```
  %% inconsistent information entries which might be hidden in a search
  %% based on (logical) consistency
for(2,nl),ttytab(20),
isa(Entity,Type),
display_p(isa(Entity,Type)),nl,
( substructures_of(Entity,Lsub),
  display_p(substructures_of(Entity,Lsub))
; true),!,
( substructures_of(SuperS,Lstr),
  memberchk(Entity,Lstr),
  display_p(substructures_of(SuperS,Lstr)),nl
; true),!,
( instances_of(Entity:_, Lsubin),
  display_p(instances_of(Entity:_, Lsubin))
; true),!,
( instances_of(SuperI,Lins),
  memberchk(Entity,Lins),
  display_p(instances_of(SuperI,Lins))
; true),!,
( process(Entity,InputL1,OutputL1,EnzymeL1,ConditionL1),
  display_p(process(Entity,InputL1,OutputL1,EnzymeL1,ConditionL1))
; true),!,
( process(Process2,InputL2,OutputL2,EnzymeL2,ConditionL2),
  memberchk(Entity,InputL2),
  display_p(process(Process2,InputL2,OutputL2,EnzymeL2,ConditionL2))
; true),!,
( process(Process3,InputL3,OutputL3,EnzymeL3,ConditionL3),
  memberchk(Entity,OutputL3),
  display_(process(Process3,InputL3,OutputL3,EnzymeL3,ConditionL3))
; true),!,
( process(Entity,InputL4,OutputL4,EnzymeL4,ConditionL4),
  memberchk(Entity,EnzymeL4),
  display_p(process(Entity,InputL4,OutputL4,EnzymeL4,ConditionL4))
; true),!,
( process(Entity,InputL5,OutputL5,EnzymeL5,ConditionL5),
  memberchk(Entity,ConditionL5),
  display_p(process(Entity,InputL5,OutputL5,EnzymeL5,ConditionL5))
; true),!,
( subprocesses_of(Entity,Subpro),
  display_p(subprocesses_of(Entity,Subpro))
; true),!,
( subprocesses_of(Superpro, SubproL),
  memberchk(Entity, SubproL),
```

```
  display_p(subprocesses_of(Superpro,Subpro))
; true),!,
( followed_by(Entity, ProcessLast),
  display_p(followed_by(Entity, ProcessLast))
; true),!,
( followed_by(ProcessFrst, Entity),
  display_p(followed_by(ProcessFrst, Entity))
; true),!,
( occurs_in(Entity, SpaceL),
  display_p(occurs_in(Entity, SpaceL))
; true),!,
% ( occurs_in(ProcessOI,Entity),
%   display_p(occurs_in(ProcessOI,Entity))
% ; true),!,
( compartment(Entity,Compartments),
  display_p(compartment(Entity,Compartments))
; true),!,
( compartment(StructC,Compartments),
  memberchk(Entity, Compartments),
  display_p(compartment(StructC,Compartments))
; true),!,
( limits_of(Entity, Lstru),
  display_p(limits_of(Entity, Lstru))
; true),!,
( limits_of(Comp, Lstru),
  memberchk(Entity, Lstru),
  display_p(limits_of(Comp, Lstru))
; true),!,
( regions_of(Entity,Lregs),
  display_p(regions_of(Entity,Lregs))
; true),!,
( regions_of(Sreg,Lregs),
  memberchk(Entity, Lregs),
  display_p(regions_of(Sreg,Lregs))
; true),!,
( neighbor_regions(Entity,Reg,Lregs2),
  display_p(neighbor_regions(Entity,Reg,Lregs2))
; true),!,
( neighbor_regions(Sreg2,Entity,Lregs3),
  display_p(neighbor_regions(Sreg2,Entity,Lregs3))
; true),!,
( neighbor_regions(Sreg3,Reg2,Lregs4),
  memberchk(Entity,Lregs4),
```

```
    display_p(neighbor_regions(Sreg3,Reg2,Lregs4))
  ; true),!,
  ( neighbor_compartments(Entity, Stl, Cin),
    display_p(neighbor_compartments(Entity, Stl, Cin))
  ; true),!,
  ( neighbor_compartments(Cout, Stl2, Entity),
    display_p(neighbor_compartments(Cout, Stl2, Entity))
  ; true),!,
  ( neighbor_compartments(Cout2, Stl3, Cin2),
    memberchk(Entity, Stl3),
    display_p(neighbor_compartments(Cout2, Stl3, Cin2))
  ; true),!,
  ( neighbor_structures(Entity, Stl4),
    display_p(neighbor_structures(Entity, Stl4))
  ; true),!,
  ( neighbor_structures(Struc, Stl5),
    memberchk(Entity,Stl5),
    display_p(neighbor_structures(Struc, Stl5))
  ; true),!,

  ask_letter('Would you like to update this',no,_Answer).


move_file(Orig, Updated):-
  readf(Orig,List),
  stamp(TimeStamp),
  writef(Updated, '%%%  session':TimeStamp),
  writelf(Updated,List),
  writef(Updated,'%%%^L%%%').

restart:-
  cls, for(5,nl), ttytab(20),
  yesno('Would you like to continue', no),
  medKAT
  ; true.

initialize_new:-
  open('new.pl', append, StrOut),
  set_output(StrOut),
  open('new.pl', read, StreamIn),
  set_input(StreamIn),
  read(StreamIn,F),
  ( F == 'end_of_file', %% If this is a NEW file
```

```
    write(StrOut, ':-multifile isa/2, instances_of/2, substructures_of/2,'),
    nl,
    write(StrOut, '            process/5, compartment/2, subprocesses_of/2,'),
    nl,
    write(StrOut, '            limits_of/2, regions_of/2, followed_by/2,'),
    nl,
    write(StrOut, '            neighbor_structures/2, neighbor_regions/3,'),
    nl,
    write(StrOut, '            neighbor_compartments/3.'),
    nl
  ; true
  ),
  seen, set_input(user), told, set_output(user).


disp_isa1(Entity,Syn,Type):-
  cls, for(2,nl), ttytab(20),
  ( var(Syn),
    display_p(isa(Entity,Type))
  ; display_p(isa(Syn, Type)),
    ttytab(10), writedq(Entity),
    write(' is also known as '), writedq(Syn),
    write(' which is used in our System.')
  ),!,
  for(10,nl), ttytab(10), writedq(Entity),
  write(' has been enterd into the system before.'),nl,
  continue.

get_type(_,Type):-
  nonvar(Type),!.
get_type(Entity,Type):-
  write('Please choose the TYPE of '), writedq(Entity),
  write(' among the following types:'),nl,
  ttytab(32), write('(1) structure'),nl,
  ttytab(32), write('(2) process'),nl,
  ttytab(32), write('(3) compartment'),nl,
  ttytab(32), write('(4) condition'),nl,
  ttytab(32), write('(5) others'),nl,
  repeat,
    ttytab(18), ask_number('Pick a number',1,5,1,Choice),
    ( check_input(Choice,[]),
      restart,
      !, fail
```

```
    ; Choice == 1,
      Type = structure
    ; Choice == 2,
      Type = process
    ; Choice == 3,
      Type = compartment
    ; Choice == 4,
      Type = condition
    ; Choice == 5,
      ask_word('Please enter type of entity',Type),
      check_input(Type,[]),
      restart,
      !, fail
    ),
  cls.

disp_isa2(Entity,Type):-
  cls, for(2,nl), ttytab(20), display_p(isa(Entity,Type)), for(5,nl).

message1(Entity,structure):-
  disp_isa2(Entity,structure),
  write('Would you like to elaborate '),writedq(Entity),
  write(' furthermore? In other words,'),nl,
  yesno('would you like to enter its SubStructures, SubClasses or SubRegions',
 no).
message1(Entity,process):-
  disp_isa2(Entity,process),
  write('Would you like to elaborate '),writedq(Entity),
  write(' furthermore? In other words,'),nl,
  write('would you like to enter its '),
  yesno('SubProcesses, SubClasses or Chained Processes',no).


message2(Process):-
  find_allsuperprocesses(Process,SuperL),
  length(SuperL,Ll),
  ( Ll =:= 0, %NO SuperProcess for Process which is a physical process
    SpaceL = [universe]
  ; Ll =:= 1,
    find_processspace(Process,SuperProcess,SpaceL),
    ( var(SuperProcess), % SpaceL is specific spacelist of Process
      list_processspace(Process,SuperL)
    ; write('There is no entry about where '),writedq(Process),
```

```
      write(' occurs;'),nl,
      write('however, we know that of '),writedq(SuperProcess),
      write(' which is its SuperProcess.'),nl,
      list_processspace(SuperProcess,SpaceL),nl,
      write('Is there any Compartment or Strucuture which is not stated'),nl,
      write('above and specifically in which '),writedq(Process),
      yesno('always occurs',no)
    )
  ; Ll > 1, %Ll has to be >0
    writedq(Process),write(' is a general process pattern'),nl,
    write('which takes part in many different proecesses such as'),nl,

    list(SuperL),nl,
    write('Therefore, it is not suitable to state a'),nl,
    write('specific Compartment in which it occurs.'),nl,
    continue,
    fail
  ).

message3(Entity,Type):-
  disp_isa2(Entity,Type),
  yesno('Would you like to provide specific attrributes for this entity',no).

message4(Entity,Known):-
  write('O.K. Let us continue on '), write(Known),
  writedq(Entity), nl.

message5(SuperClass:Scope, Entity):-
  write('Is '),writedq(Entity),write(' A/AN '),writedq(SuperClass:Scope),
  yesno('',yes);
  write('Then there is no Sub-SuperClass relation.'),nl, fail.
message6(Entity):-
  write('Please enter substrates of the process one by one'),nl,
  write('SUBSTRATES of '), write(Entity), nl.
message7(Entity):-
  write('Please enter products of the process one by one'),nl,
  write('PRODUCTS of '), write(Entity), nl.
message8(Entity):-
  write('Please enter enzymes of the process one by one'),nl,
  write('ENZYMES of '), write(Entity), nl.
message9(Entity):-
  write('Please enter conditions of the process one by one'),nl,
  write('CONDITIONS of '), write(Entity), nl.
```

```
message10(Entity,structure,SuperL):-
  consult('new.pl'),
  disp_isa2(Entity,structure),
  ( find_superstructure(Entity,_),
    list_superstructures(Entity),nl,!,
    yesno('Would you like to add more to this (SuperStructure) list',no)
  ; write('Is there any Structure you know where '),writedq(Entity),
    write(' always is a part of which?'),nl,
    write('If you do not know the answer press enter.'),nl,nl,
    writedq(Entity),write(' is a SubStructure of ')
  ),
  ask_list(SuperL:_,1),
  ( member(Super,SuperL),
      writedq(Entity),write(' is NOT A/AN '),writedq(Super),
      \+ yesno('but an "essential structural part" of it',yes),
      !,fail
  ; true
  ).
message10(Entity,process,SuperL):-
  consult('new.pl'),
  disp_isa2(Entity,process),
  ( find_superprocess(Entity,_),
    list_superprocesses(Entity),nl,!,
    yesno('Would you like to add more to this (SuperProcess) list',no)
  ; write('Is there any Process you know where '),writedq(Entity),
    write(' always is a part of which?'),nl,
    write('If you do not know the answer press enter.'),nl,nl,
    writedq(Entity),write(' is a SubProcess of ')
  ),
  ask_list(SuperL:_,1),
  ( member(Super,SuperL),
      writedq(Entity),write(' is NOT A/AN '),writedq(Super),
      \+ yesno('but an "essential functional part" of it',yes),
      !,fail
  ; true
  ).
message11(Entity,Type):-
  consult('new.pl'),
  disp_isa2(Entity,Type),
  ( find_superclass(Entity,_),
    list_superclasses(Entity),nl,!,
    yesno('Would you like to add more to this (SuperClass) list',no)
  ; write('What is '),writedq(Entity),write('?'),nl,
```

```
    write('If you do not know the answer press enter.'),nl,
    writedq(Entity),write(' is a'),nl
  ).
message12(Entity,structure):-
  consult('new.pl'),
  disp_isa2(Entity,structure),
  ( find_substructure(_,Entity),!,
    list_substructures(Entity)
  ; yesno('Do you know its SubStructure',no)
  ),
  write('Please press enter when you wish to finish.'),nl,
  write('SubStructures of '),writedq(Entity), write(' are,'),nl.
message12(Entity,process):-
  consult('new.pl'),
  disp_isa2(Entity,process),
  ( find_subprocess(_,Entity),
    list_subprocesses(Entity)
  ; yesno('Do you know its SubProcesses',no)
  ),
  write('Please press enter when you wish to finish.'),nl,
  write('SubProcesses of '),writedq(Entity), write(' are,'),nl.
message13(Entity,Type):-
  consult('new.pl'),
  disp_isa2(Entity,Type),
  ( find_subclass(_,Entity:_),!,
    write('Followings are also known as '),writedq(Entity), nl,
    list_subclasses(Entity)
  ; write('Is there any '),write(Type), write(' you know, which is a/an '),
    writedq(Entity), yesno('',no)
  ),
  write('The '),write(Entity),write('s we know are,').
message14(Entity,Type,Path):-
  disp_isa2(Entity,Type),
  write(Entity), write(' has been integrated within the network.'),nl,
  write('Its path to the root is : '),
  write(Path),nl,
  continue.
message15(Entity,Type,Path):-
  disp_isa2(Entity,Type),
  write(Entity), write(' could not be rooted properly.'),nl,
  write(Path),nl,
  write('You can root the new info via defining its SuperStructure'),nl,
  write('and/or SuperClass when they are asked on screen.'),nl,
```

```
  continue,
  fail.
message16(Class:Scope,Scope4,Entity):-
  ( nonvar(Scope), ClassS = Class:Scope
  ; ClassS = Class
  ),!,
  ttytab(3),
  write('One of '),writedq(ClassS),write(' of a particular organism or'),nl,
  ttytab(3),
  write('in a particular environment is '), writedq(Entity),nl,
  ttytab(3),
  write('If this is ALWAYS true then press enter else write its specific '),nl,
  ttytab(3),
  write('scope which is the "particular" SubClass of '),
  (nonvar(Scope), writedq(Class:Scope)
  ; writedq(Class)
  ), write('.'),nl,

  ( nonvar(Scope), ask_word(Class:Scope,S)
  ; ask_word(Class,S)
  ),
  ( check_input(S,[]),
    Scope4 = all
  ; ( var(Scope), Scope2 = S
    ; Scope2 = Scope:S
    ),!,
    ttytab(10),write('instances_of('),write(Class:Scope2),
    write(', ['),write(Entity),write(']).'),nl,
    message16(Class:Scope2,Scope3,Entity),
    Scope4 = Scope2:Scope3
  ),!.

message18(Entity,structure):-
  disp_isa2(Entity,structure),
  write('If '),writedq(Entity),write(' cannot be described straightforward'),nl,
  write('you can partition it and start to describe from that point on.'),nl,
  write('But this should not have structure-substructure relation, which'),nl,
  write('should be done in earlier stages. If the case is that, please'),nl,
  write('answer no, and redo after finishing this process. Thank You!'),nl,
  write('Is partitioning suitable? y/n : ').

message19(Entity,structure):-
  disp_isa2(Entity,structure),
```

```
  write('Please enter the major parts of '), writedq(Entity),nl,
  write('regardless their describability. It is most suitable to '),nl,
  write('partition it in as few as possible parts first, and'),nl,
  write('partition them into subparts recursively. Please start to do.'),nl,
  for(2,nl).

list_substructures(Entity):-
  write('According to the current information in our System'),nl,nl,
  findall(Sub,find_substructure(Sub,Entity),SubL),
  write('SubStructures of '), writedq(Entity), write(' are:'),nl,
  write('============='),nl,
  list(SubL).

list_subprocesses(Entity):-
  write('According to the current information in our System'),nl,nl,
  findall(Sub,find_subprocess(Sub,Entity),SubL),
  write('SubProcesses of '), writedq(Entity), write(' are:'),nl,
  write('============'),nl,
  list(SubL).

list_subclasses(Entity):-
  findall(Sub,find_subclass(Sub,Entity:_),SubL),
  write('SubClasses of '), writedq(Entity), write(' are:'),nl,
  write('=========='),nl,
  list(SubL).

list_superstructures(Entity):-
  write('According to current information in our model'),nl,nl,
  findall(Super,find_superstructures(Entity, Super),SuperL),
  list_to_set(SuperL,SuperSet),
  write('SuperStructures of '), writedq(Entity), write(' are:'),nl,
  write('==============='),nl,
  list(SuperSet).

list_superprocesses(Entity):-
  write('According to the current information in our System'),nl,nl,
  findall(Super,find_superprocesses(Entity, Super),SuperL),
  list_to_set(SuperL,SuperSet),
  write('SuperProcesses of '), writedq(Entity), write(' are:'),nl,
  write('=============='),nl,
  list(SuperSet).

list_superclasses(Entity):-
```

```
  write('According to current information in our model '),nl,
  findall(Super,find_superclasses(Entity, Super),SuperL),
  list_to_set(SuperL,SuperSet),
  writedq(Entity),write(' is considered as,'),nl,
  list(SuperSet).

list_processspace(Process,SpaceL):-
  write('Compartments and Structures where '),writedq(Process),
  write(' occurs are as follows,'),nl,
  list(SpaceL).

list(List):-
  list2(List,1).
list2([],_):-!.
list2([H|T],C):-
  ( C<10, ttytab(1); true),
  write(C),write('.'),
  write(H),nl,
  C1 is C + 1,
  list2(T,C1).


                          %%%%%%%%%%%%%%%%%%
                          %%%% mylib.pl %%%%%
                          %%%%%%%%%%%%%%%%%%

:-[library(basics), library(lists), library(sets), library(date),
   library(prompt), library(ask), library(not)].


cls:-
  \+ unix(system(clear)).

%% C: Counter
for(C, Procedure):-
  next(C,1,Procedure).
next(C,C0,_):-
  C0 > C, !.
next(C,C0,Procedure):-
  Procedure,
  C1 = C0 +1,
  next(C,C1,Procedure).
```

```
reading(Answer):-
  read(Answer),
  ( Answer == q, abort
  ; true
  ).

check_input(Item,List_end):-
  ( Item == ''
  ; Item == ' '
  ; Item == 'n'
  ; Item == 'N'
  ; Item == 'q'
  ; Item == 'e'
  ; Item == []
  ),
  List_end = [].

check_list_subparts([],[],_Super).
check_list_subparts(List,ListRevised,Super):-
  List = [Sub|Tail],
  write('Is '),writedq(Sub),write(' NOT A/AN '),writedq(Super),nl,
  ( yesno('but a part of it',yes),
    ListRevised = [Sub|Hole]
  ; ListRevised = Hole
  ),!,
  check_list_subparts(Tail,Hole,Super).

check_list_subclasses([],[],_Super).
check_list_subclasses(List,ListRevised,Super):-
  List = [Sub|Tail],
  write('Is '),writedq(Sub),write(' is A/AN '),writedq(Super),
  ( yesno('',yes),
    ListRevised = [Sub|Hole]
  ; ListRevised = Hole
  ),!,
  check_list_subclasses(Tail,Hole,Super).
%check_list_classes(List,ListRevised,Sub):-

ask_letter(Prompt,Default,Answer):-
  ask(Prompt,Default,Ans),
  atom_chars(Answer,[Ans]).
ask_word(Prompt,Answer):-     %% plain enter means ''
  ask_chars(Prompt,0,250,Ans),
```

```
  atom_chars(Answer,Ans).
ask_query(Prompt,Answer):-    %% plain enter means []
  ask_chars(Prompt,0,250,Ans),
  ( Ans == [] ,
    Answer = Ans
  ; atom_chars(Answer,Ans)
  ),!.
ask_list(L:T, C):-
  ask_query(C, Substance),
  ( check_input(Substance,[]),!,
    L = Substance
  ; L = [Substance|T],
    C1 is (C + 1),
    ask_list(T:_T1, C1)
  ).

ask_superclasses(Entity,L:T, C):-
  ask_word(C, Class),
  ( check_input(Class,L),!
  ;
    message16(Class:_,Scope,Entity),nl,
    ( message5(Class:Scope,Entity),
      L = [Class|T],
      writef('new.pl',instances_of(Class:Scope,[Entity])),
      C1 is (C + 1)
    ; L = T,
      C1 is C
    ),!,
    ask_superclasses(Entity,T:_T1, C1)
  ).

ask_subclasses(Entity,L:T, C):-
  ask_word(C, Class),
  ( check_input(Class,L),!
  ;
    L = [Class|T],
    C1 is (C + 1),
    ask_subclasses(Entity,T:_T1, C1)
  ).

continue:-
  nl,
  prompted_char('Please press enter to continue... ',_).
```

```
display_p(Predicate):-
       Predicate =..L,
       L=[H|T],
       write(H),write('('),
       T=[Th|Tt],
       write(Th),
       member(A,Tt),
         write(','),
         write(A),
 fail
       ; write(').'),nl.

error_message:- write('!! ERROR !! : Inconsistent Knowledge Entry'),nl,
        write('========================================'),nl.

%% used for searching for a word which is not used within database
%% synonymous([M|L]): M is the Main word used in db, so L is searched
syn_used(X,M,T):-
  synonymous([M|L]),
  memberchk(X,L), !,
  isa(M,T).

list_predicates(StreamIn, L:T:Tn):-
  read(StreamIn, P),
  L=[P|T],
  \+ P == 'end_of_file',
  list_predicates(StreamIn, T:_T1:Tn)
  ; L = [],!.

find_substructure(Sub,Super):-
  substructures_of(Super,SubL),
  member(Sub,SubL).
find_subprocess(Sub,Super):-
  subprocesses_of(Super,SubL),
  member(Sub,SubL).
find_subclass(Sub,Super):-
  instances_of(Super,SubL),
  member(Sub,SubL).

find_superstructure(Sub, Super):-
  substructures_of(Super,List),
  memberchk(Sub,List).
```

```
find_superprocess(Sub, Super):-
  subprocesses_of(Super,List),
  memberchk(Sub,List).
find_superclass(Sub, Super):-
  instances_of(Super:_,L),
  memberchk(Sub,L).

find_superclasses(Sub,Super):-
  ( find_superclass(Sub,Super)
  ;
    find_superclass(Sub,Sup),
    find_superclasses(Sup,Super)
  ).
find_superstructures(Sub,Super):-
  ( find_superstructure(Sub,Super)
  ; find_superclasses(Sub,SuperClass),
    find_superstructure(SuperClass,Super)
  ).
find_superprocesses(Sub,Super):-
  ( find_superprocess(Sub,Super)
  ; find_superclasses(Sub,SuperClass),
    find_superprocess(SuperClass,Super)
  ).

find_allsuperclasses(Sub,SuperL):-
  findall(Super,find_superclasses(Sub,Super),List),
  list_to_set(List,SuperL).
find_allsuperstructures(Sub,SuperL):-
  findall(Super,find_superstructures(Sub,Super),List1),
  find_allsuperclasses(Sub,ClassL),
  findall(Super, (member(C,ClassL),find_superstructures(C,Super)), List2),
  ( List1 \== [],
    List2 \== [],
    append(List1,List2,List)
  ; List1 \== [],
    List = List1
  ; List = List2
  ),
  list_to_set(List,SuperL).
find_allsuperprocesses(Sub,SuperL):-
  findall(Super,find_superprocesses(Sub,Super),List1),
  find_allsuperclasses(Sub,ClassL),
  findall(Super, (member(C,ClassL),find_superprocesses(C,Super)), List2),
```

```
  ( List1 \== [],
    List2 \== [],!,
    append(List1,List2,List)
  ; List1 \== [],!,
    List = List1
  ; List = List2
  ),
  list_to_set(List,SuperL).

find_processspace(Process,SuperProcess,SpaceL):-
  ( occurs_in(Process,SpaceL),!,
    SuperProcess = Process
  ;
    find_superprocesses(Process,Super),
    find_processspace(Super,SuperProcess,SpaceL)
  ).

is_there(Entity,_,Type):-
  consult('new.pl'),
  isa(Entity,Type),!.
is_there(Entity,Syn,Type):-
  syn_used(Entity,Syn,Type).

writef(File, Predicate):-
  open(File, append, StreamOut),
  set_output(StreamOut),
  write_term(StreamOut, Predicate,[quoted(true)]),
  write('.'), nl,
  told,
  set_output(user).
writelf(File, ListOfPredicates):-
  open(File, append, StreamOut),
  set_output(StreamOut),
  repeat,
    ( member(Predicate, ListOfPredicates),
        % write_term(StreamOut, Predicate,[quoted(true)]), write('.'), nl,
        write_term(StreamOut, Predicate), write('.'), nl,
      fail
    ; true,!
    ),
  told,
  set_output(user).
writedq(Entity):-
```

```
  write('"'),
  write(Entity),
  write('"').

readf(File,Ls):-
  open(File, read, StreamIn),
  set_input(StreamIn),
  read(StreamIn,_MULTIFILE_LineIntoJUNK),
  list_predicates(StreamIn, L:_),
  seen, set_input(user),
  findall(isa(E,T), member(isa(E,T),L), Lisa),
  subseq(L, Lisa, Lno_isa),
    %% Lno_isa contains everything but predicates of isa
  sort(Lno_isa, Lno_isa_s1),
  sort(Lisa, Lisa_s1),
  setof(X-isa(E,X), member(isa(E,X),Lisa_s1), Lisa_s2),
    %% setof also sorts automatically
  findall(isa(E,T),member(T-isa(E,T),Lisa_s2), Lisa_s3),
  append(Lisa_s3, Lno_isa_s1,Ls), !.

stamp(TimeStamp):-
  datime(X),
  X = date(Yr,Mo,Da,Hr,Min,_),
  TimeStamp =  Mo/Da/Yr - Hr:Min.

rooted(S,structure):-
  substructures_of(So,L),
  memberchk(S,L),!,
  rooted(So,structure).
rooted(P,process):-
  subprocesses_of(Po,L),
  memberchk(P,L),!,
  rooted(Po,process).
rooted(S,Type):-
  instances_of(So:_,L),
  memberchk(S,L),!,
  rooted(So,Type).
rooted(E,Type):-
  Type == structure,
    ( E == ecosystem,!
    ; E == last:structure,
      write('No inconsistency found in the predicate substructures_of/2'),nl
    )
```

```
  ;
  Type == process,
    ( ( E == tca_cycle ; E == pyruvate_dehydrogenization), !
    ; E == last:process,
    write('No inconsistency found in the predicate subprocesses_of/2'),nl
    )
  ;
  error_message,
  fail,!.


                              %%%%%%%%%%%%%%%%%%
                              %%%%% syn.pl %%%%%%
                              %%%%%%%%%%%%%%%%%%


%% This file contains facts called synonymous and clause called synonyms
%% which check whether there is a synonym of a word entered before

%% synonymous(+List) consists of synonymous words

synonymous([nicotinic_acid, nicotinate, vitamin_B3]).
synonymous([pantothenic_acid, pantothenate]).
synonymous([reactant, substrate]).
synonymous([intracellular_space, intracellular_compartment]).
synonymous([extracellular_space, extracellular_compartment]).
synonymous([tca_cycle,
     tricarboxylic_acid_cycle,
   citric_acid_cycle,
   krebs_cycle]).
synonymous([cellular_process, intracellular_process]).
synonymous([metabolic_processes, metabolism]).
synonymous([cytosol,hyaloplasm,cell_sap]).
synonymous([ectoplasm, cell_cortex]).
synonymous([plasma_membrane,cell_membrane]).
synonymous([er,endoplasmic_reticulum,cisternae]).
synonymous([ser, smooth_endoplasmic_reticulum,
    agranular_er, agranular_endoplasmic_reticulum]).
synonymous([ger,granular_endoplasmic_reticulum,
    rer,rough_endoplasmic_reticulum]).
synonymous([golgi_complex,golgi, golgi_apparatus,golgi_body,golgi_structure]).
synonymous([dictyosome,golgi_sac,golgi_saccule]).
synonymous([peroxisome, microbody]).
```

```
synonymous([mesosome, chondrioid]).
synonymous([lamellate_cytomembrane, lamellated_cytomembrane]).
synonymous([scretory_material,secretion_material,secrete,secretion]).
synonymous([pyruvate_dehydrogenase, pdh]).
synonymous([alpha_ketoglutarate_dehydrogenase_complex,
    alpha_ketoglutarate_dehydrogenase]).
synonymous([acetyl_CoA, acetyl_coenzyme_A, acetyl_coenzyme_a]).
synonymous(['CoASH', 'CoA', coenzyme_A, coenzyme_a]).
synonymous([gdp, 'GDP', guanosine_diphosphate]).
synonymous([gtp, 'GTP', guanosine_triphosphate]).
synonymous(['P',pi,'Pi',phosphate_radical]).
synonymous([thiamine_pyrophosphate, 'TPP']).
synonymous([thiamine, vitamin_B1]).
synonymous([fumarase,fumarate_hydratase]).
synonymous([oxaloacetate,'oxaloacetate^2-','oxaloacetate--']).
synonymous([citrate,'citrate^3-',citric_acid]).
synonymous([citrate_synthesis,citric_acid_synthesis]).
synonymous([citrate_synthase,'citrate_oxaloacetate_lyase (CoA-acetylating)',
    condensing_enzyme, acetyle_CoA_oxaloacetate_condensing_enzyme,
    citrate_condensing_enzyme, oxaloacetate_transacetase, citrogenase,
    citrate_synthetase]).

%% synonyms(?Word, ?ItsSynonymous) searches Word within predicate synonymous
%% responds with all synonyms of Word but Word itself.
synonyms(X,Y):-
synonymous(SynList),
memberchk(X, SynList),!,
member(Y, SynList),
  X \= Y.


                          %%%%%%%%%%%%%%%%%%
                          %%%%% isa.pl %%%%%
                          %%%%%%%%%%%%%%%%%%
:-multifile isa/2.

%% structures

isa(input_substance,structure).
isa(substance,structure).
isa(atom,structure).
isa(proton,structure).
isa(electron,structure).
```

```
isa(chemical_process_substrate,structure).
isa(substrate,structure).
isa(chemical_process_product,structure).
isa(product,structure).
isa(biochemical_process_substrate,structure).
isa(biochemical_process_product,structure).
isa(enzyme,structure).
isa(control_feature,structure).
isa(acetyl,structure).
isa(beta_mercapto_ethylamine_residue,structure).
isa(sulfhydryl,structure).
isa('H',structure).
isa(ethyl,structure).
isa('S',structure).
isa(thioester_bond,structure).
isa(citrate_synthase_oxaloacetate_complex,structure).
isa(tca_cycle_substance,structure).
isa(thioenolate_anion,structure).
isa(biochemical_substance,structure).
isa(organic_substance,structure).
isa(oxaloacetate,structure).
isa(bacterium,structure).
isa(blue_green_alga,structure).
isa(peroxisomal_membrane,structure).
isa(ribosomal_membrane,structure).
isa(ribosome,structure).
isa(lysosome,structure).
isa(peroxisome,structure).
isa(free_secretory_vesicle_membrane,structure).
isa(free_transfer_vesicle_membrane,structure).
isa(prokaryotic_cell,structure).
isa(eukaryotic_cell,structure).
isa(vacuolar_membrane,structure).
isa(er_membrane,structure).
isa(lysosomal_membrane,structure).
isa(ecosystem,structure).
isa(organism ,structure).
isa(physiological_system,structure).
isa(organ,structure).
isa(cell,structure).
isa(connective_tissue,structure).
isa(peripheral_artery,structure).
isa(peripheral_vena,structure).
```

```
isa(peripheral_lymph_vessel,structure).
isa(peripheral_nerve_cell,structure).
isa(membrane,structure).
isa(plasma_membrane,structure).
isa(cytoplasm,structure).
isa(organelle,structure).
isa(organelle_content, structure).
isa(cytosol,structure).
isa(cytoskeleton,structure).
isa(cytosol_content,structure).
isa(er,structure).
isa(ger,structure).
isa(ser,structure).
isa(mitochondrion,structure).
isa(mitochondrial_outer_membrane,structure).
isa(intermembrane_space_content,structure).
isa(mitochondrial_inner_membrane,structure).
isa(mitochondrial_matrix_substance,structure).
isa(mitochondrial_DNA,structure).
isa(mitochondrial_RNA,structure).
isa(pyruvate_dehydrogenase_enzyme_complex, structure).
isa(tca_cycle_enzyme,structure).
isa(citrate,structure).
isa(citrate_synthase,structure).
isa(isocitrate,structure).
isa(isocitrate_dehydrogenase,structure).
isa(aconitase,structure).
isa(alpha_ketoglutarate,structure).
isa(alpha_ketoglutarate_dehydrogenase_complex,structure).
isa(succinyl_CoA,structure).
isa(succinyl_CoA_synthetase,structure).
isa(succinate,structure).
isa(succinate_dehydrogenase,structure).
isa(fumarate,structure).
isa(fumarase,structure).
isa(l_malate,structure).
isa(malate_dehydrogenase,structure).
isa(fatty_acid,structure).
isa(fatty_acyl_CoA,structure).
isa(acetyl_CoA,structure).
isa(pyruvate,structure).
isa(thiamine,structure).
isa(thiamine_pyrophosphate,structure).
```

```
isa('CoASH',structure).
isa('ADP',structure).
isa('ATP',structure).
isa('GDP',structure).
isa('GTP',structure).
isa('Pi',structure).
isa('NAD+',structure).
isa('NADH',structure).
isa('FAD',structure).
isa('FADH2',structure).
isa('H+',structure).
isa('O2',structure).
isa('H2O',structure).
isa('CO2',structure).
isa('Fe++',structure).
isa('Ca++',structure).
isa('Mg++',structure).
isa(golgi_complex,structure).
isa(dictyosome,structure).
isa(free_secretory_vesicle,structure).
isa(free_transfer_vesicle,structure).
isa(golgi_membrane,structure).
isa(free_inorganic_cytosol_material,structure).
isa(organic_cytosol_material, structure).
isa(secretory_material,structure).
isa(vesicle,structure).
isa(vesicular_membrane,structure).
isa(vesicular_material,structure).
isa(nucleus,structure).
isa(nucleolus,structure).
isa(nuclear_envelope,structure).
isa(nucleoplasm,structure).
isa(chromatin,structure).
isa(outer_nuclear_membrane,structure).
isa(inner_nuclear_membrane,structure).
isa(nuclear_envelope_pores,structure).
isa(cytoskeleton,structure).
isa(microfilament,structure).
isa(microtubule,structure).
isa(intermediate_filament,structure).
isa(animal_cell,structure).
isa(plant_cell,structure).
isa(escherichia_coli,structure).
```

```
isa(centriole,structure).
isa(phycobilosome,structure).
isa(flagellum,structure).
isa(plastid,structure).
isa(cell_wall,structure).
isa(vacuole,structure).
isa(lamellate_cytomembrane,structure).
isa(mesosome,structure).
isa(nucleoid,structure).
isa(circular_dna,structure).
isa(pplo,structure).
isa(virus,structure).
isa(last:structure,structure). %% Mark for end

%% compartments

isa(intracellular_space,compartment).
isa(intermembrane_space,compartment).
isa(mitochondrial_matrix,compartment).
isa(last:compartment,compartment). %% Mark for end

%% processes

isa(chemical_process,process).
isa(biochemical_process,process).
isa(physical_process,process).
isa(pyruvate_dehydrogenization,process).
isa(tca_cycle,process).
isa(citrate_synthesis,process).
isa(citrate_synthesis_step1,process).
isa(isocitrate_synthesis,process).
isa(isocitrate_dehydrogenization,process).
isa(alpha_ketoglutarate_dehydrogenization,process).
isa(succinyl_CoA_synthesis,process).
isa(succinate_dehydrogenization,process).
isa(fumarate_hydratization,process).
isa(malate_dehydrogenization,process).
isa(citrate_dehydratization,process).
isa(cis_aconitate_hydratization,process).
isa(last:process,process).  %% Mark for end


%% others
```

```
isa(dG:_, energy).
isa(all, quantifier).


                              %%%%%%%%%%%%%%%%%%%%%
                              %%%% structure.pl %%%%
                              %%%%%%%%%%%%%%%%%%%%%
:-multifile substructures_of/2, granul/2.

%structural decomposition FACTS

%substructures of particular system are given as a list
%substructures_of(system, [list of substructures])

substructures_of(ecosystem,[organism] ).
substructures_of(ecosystem,[input_substance]).
substructures_of(ecosystem,[substance]).
substructures_of(substance,[element,molecule]).
substructures_of(element,[atom,molecule]).
substructures_of(molecule,[atom]).
substructures_of(atom,[proton,electron]).
substructures_of(organism, [cell]).
substructures_of(complex_organism,[physiological_system,organ]).
substructures_of(organ,[cell,
connective_tissue,
peripheral_artery,
peripheral_vena,
peripheral_lymph_vessel,
peripheral_nerve_cell]).
substructures_of(cell, [plasma_membrane,cytoplasm]). %minimal description
substructures_of(cytoplasm, [organelle, cytosol]).
substructures_of(cytosol, [cytoskeleton,
   cytosol_content]).
substructures_of(cytosol_content,[vesicle,
  free_inorganic_cytosol_material,
  organic_cytosol_material]).
substructures_of(organelle, [membrane, organelle_content]).
substructures_of(er, [ger,ser]).
substructures_of(mitochondrion,[mitochondrial_outer_membrane,
intermembrane_space_content,
               mitochondrial_inner_membrane,
               mitochondrial_matrix_substance]).
```

```
substructures_of(golgi_complex, [dictyosome,
 free_secretory_vesicle,
 free_transfer_vesicle]).
substructures_of(dictyosome, [golgi_membrane, secretory_material]).
substructures_of(vesicle,[vesicular_membrane, vesicular_material]).

%% The direction of this clause needs depth of biomedical information
%% which would take too much time though it is clinically very important
%% substructures_of(secretory_material, List):-
%%              query(secretory_material:golgi_complex, CellType),
%%              (   CellType == xyz -> List = [x1,x2,x3]
%%              ;   CellType == xyy -> List = [y1,y2,y3,y4]).

substructures_of(nucleus, [nucleolus,
                  nuclear_envelope,
                  nucleoplasm,
                  chromatin]).
substructures_of(nuclear_envelope, [outer_nuclear_membrane,
           inner_nuclear_membrane,
           nuclear_envelope_pores]).
substructures_of(cytoskeleton, [microfilament,
                microtubule,
                intermediate_filament]).
substructures_of(acetyl_CoA,[thioester_bond]).
substructures_of(thioester_bond,['S',carbonyl]).
substructures_of(acetyl_CoA,[acetyl,'CoA']).
substructures_of('CoASH',[beta_mercapto_ethylamine_residue,
                        pantothenic_acid_residue,
                        'ADP_with_phosphate_at_C3']).
substructures_of(beta_mercapto_ethylamine_residue,[sulfhydryl,ethyl,amine]).
substructures_of(sulfhydryl,['S','H']).

granul(tca_cycle_substance,molecular).


                          %%%%%%%%%%%%%%%%%%%%%%%%
                          %%%%% instance.pl %%%%%%%
                          %%%%%%%%%%%%%%%%%%%%%%%%
:-multifile instances_of/2.

%actual instances of an entity (structure or process)
%of which characteristics are inherited from that entity
%instances_of(entity, [list_of_instances]).
```

```
instances_of(physical_process:all,[natural_process]).
instances_of(substance:all,
             [input_substance,output_substance,control_substance]).
instances_of(input_substance:all,[substrate]).
instances_of(output_substance:all,[product]).
instances_of(control_substance:macromolecular:all,[enzyme]).
instances_of(substrate:all,[chemical_process_substrate]).
instances_of(product:all,[chemical_process_product]).
instances_of(chemical_process_substrate:all,[biochemical_process_substrate]).
instances_of(chemical_process_product:all,[biochemical_process_product]).

instances_of(organism:all,[complex_organism,primitive_organism]).
instances_of(cell:all,[eukaryotic_cell, prokaryotic_cell]).
instances_of(eukaryotic_cell:all,[animal_cell, plant_cell]).
instances_of(prokaryotic_cell:all,[bacterium, blue_green_alga, pplo, virus]).
instances_of(bacterium:all, [escherichia_coli]).
instances_of(membrane:all,[plasma_membrane,
                           nuclear_envelope,
   vacuolar_membrane,
   mitochondrial_inner_membrane,
   mitochondrial_outer_membrane,
   lysosomal_membrane,
   peroxisomal_membrane,
   ribosomal_membrane,
   er_membrane,
   golgi_membrane,
   free_secretory_vesicle_membrane,
   free_transfer_vesicle_membrane]).
instances_of(organelle:all,[nucleus,
    ribosome,
    er,
    golgi_complex,
    mitochondrion,
    plastid,
    lysosome,
    peroxisome,
    centriole]).
instances_of(organelle:eukaryotic_cell, [nucleus,
                mitochondrion,
                lysosome,
                peroxisome,
```

```
                er,
                golgi_complex]).
instances_of(organelle:animal_cell, [centriole]).
instances_of(organelle:plant_cell, [cell_wall,
    vacuole,
    plastid]).
instances_of(organelle:virus, []).
instances_of(organelle:pplo, [circular_dna,
      ribosome]).
instances_of(organelle:bacterium, [nucleoid,
   ribosome,
          mesosome,
          lamellate_cytomembrane,
          flagellum]).
instances_of(organelle:blue_green_alga, [nucleoid,
                ribosome,
                phycobilosome]).
instances_of(vesicle:cell:all,[free_secretory_vesicle,
      free_transfer_vesicle,
      dictyosome,
      ribosome,
      peroxisome,
      lysosome,
      vacuole]).
instances_of(vesicular_membrane:cell,[vacuolar_membrane,
  lysosomal_membrane,
  peroxisomal_membrane,
  ribosomal_membrane,
  free_secretory_vesicle_membrane,
  free_transfer_vesicle_membrane,
  golgi_membrane]).
instances_of(mitochondrial_matrix_substance:all,
 [mitochondrial_DNA,
  mitochondrial_RNA,
  tca_cycle_substance,
  fatty_acid,
  fatty_acyl_CoA,
  pyruvate,
  pyruvate_dehydrogenase_enzyme_complex,
  'ADP',
  'ATP',
  'GDP',
  'GTP',
```

```
  thiamine,
  thiamine_pyrophosphate,
  'Fe++',
  'Ca++',
  'Mg++',
  'hundreds of other enzymes(no info available)',
  'many other organic molecules (no info available)',
  'many other inorganic molecules(no info available)']).
instances_of(tca_cycle_substance:all, [tca_cycle_substrate,
               tca_cycle_product,
               tca_cycle_enzyme]).
instances_of(tca_cycle_substrate:cell,[oxaloacetate,
                                       citrate,
       isocitrate,
       alpha_ketoglutarate,
             succinyl_CoA,
       succinate,
       fumarate,
       l_malate,
       'CoASH',
       acetyl_CoA,
       'GDP',
       'Pi',
       'NAD+',
       'FAD',
       'O2',
       'H2O']).
instances_of(tca_cycle_product:cell,[oxaloacetate,
     citrate,
     isocitrate,
     alpha_ketoglutarate,
     succinyl_CoA,
     succinate,
     fumarate,
     l_malate,
     'CoASH',
     'NADH',
     'H+',
     'CO2',
     'GTP',
     'FADH2']).
instances_of(tca_cycle_enzyme:all,[citrate_synthase,
                                   aconitase,
```

```
                                isocitrate_dehydrogenase,
                                alpha_ketoglutarate_dehydrogenase_complex,
                                succinyl_CoA_synthetase,
                                succinate_dehydrogenase,
                                fumarase,
                                malate_dehydrogenase]).
instances_of(tca_cycle_substance:all,[citrate_synthase_oxaloacetate_complex]).
instances_of(tca_cycle_substance:all,[thioenolate_anion]).
instances_of(resonance_stabilized_carbanion:all,[thioenolate_anion]).
instances_of(biochemical_substance:eukaryotic_cell:all,[tca_cycle_substance]).
instances_of(organic_substance:all,[biochemical_substance]).


                            %%%%%%%%%%%%%%%%%%%%%%%%
                            %%%%%% process.pl %%%%%%%
                            %%%%%%%%%%%%%%%%%%%%%%%%
:-multifile process/5.
:-multifile subprocesses_of/2.
:-multifile followed_by/2.
:-multifile occurs_in/2.

%% process(Name,
%%         InputList, %% Substrates
%%         OutputList,%% Products
%%         EnzymeList,
%%         ConditionList).%% essential inorganic material,
  %% rate of reactions (k1,k-1,k2,k-2,Km,Vmax,vf
  %% concentrations of materials [S],[E],[ES],[P]

%% subprocesses_of(ProcessName, SuprocessList).

%% followed_by(Process1, Process2).

%% occurs_in(processName, Compartment(Structure)List).

process(physical_process,
[input_substance],[output_substance],[control_feature],['T > 0']).
process(chemical_process,
[chemical_process_substrate],
        [chemical_process_product],[],[]).
process(biochemical_process,
[biochemical_process_substrate],
[biochemical_process_product],
```

```
[enzyme],
[living_organismal_condition]).
process(pyruvate_dehydrogenization,
        [pyruvate, 'NAD+', 'CoASH'],
[acetyl_CoA, 'CO2', 'NADH', 'H+'],
[pyruvate_dehydrogenase_enzyme_complex],
[dG: -8]).
process(citrate_synthesis,
[acetyl_CoA, oxaloacetate, 'H2O'],
[citrate, 'CoASH', 'H+'],
[citrate_synthase],
[dG : -9.08]).
process(isocitrate_synthesis,
[citrate],
[isocitrate],
[aconitase],
[dG: 3.18,'Fe++']).
process(isocitrate_dehydrogenization,
[isocitrate, 'NAD+'],
[alpha_ketoglutarate, 'NADH','CO2','H+'],
[isocitrate_dehydrogenase],
[dG: -1.7]).
process(alpha_ketoglutarate_dehydrogenization,
[alpha_ketoglutarate, 'NAD+','CoASH'],
[succinyl_CoA,'NADH','H+','CO2'],
[alpha_ketoglutarate_dehydrogenase_complex],
[dG: -8.82]).
process(succinyl_CoA_synthesis,
[succinyl_CoA, 'GDP', 'Pi'],
[succinate, 'GTP', 'CoASH'],
[succinyl_CoA_synthetase],
[dG: -2.12]).
process(succinate_dehydrogenization,
[succinate, 'FAD'],
[fumarate,'FADH2'],
[succinate_dehydrogenase],
[dG : 0]).
process(fumarate_hydratization,
[fumarate, 'H2O'],
[l_malate],
[fumarase],
[dG : -0.88]).
process(malate_dehydrogenization,
```

```
[l_malate,'NAD+'],
[oxaloacetate, 'NADH','H+'],
[malate_dehydrogenase],
[dG : 6.69]).

process(citrate_synthesis_step1,
[citrate_synthase,acetyl_CoA,oxaloacetate],
[citrate_synthase_oxaloacetate_complex,thioenolate_anion],
[],[]).

subprocesses_of(natural_process,[live,chemical_process]).
subprocesses_of(live,[biochemical_process]).
subprocesses_of(tca_cycle,
               [citrate_synthesis,
                isocitrate_synthesis,
                isocitrate_dehydrogenization,
                alpha_ketoglutarate_dehydrogenization,
                succinyl_CoA_synthesis,
                succinate_dehydrogenization,
                fumarate_hydratization,
                malate_dehydrogenization]).
subprocesses_of(isocitrate_synthesis,
  [citrate_dehydratization,
 cis_aconitate_hydratization]).

followed_by(pyruvate_dehydrogenization, citrate_synthesis).
followed_by(citrate_synthesis, isocitrate_synthesis).
followed_by(isocitrate_synthesis, isocitrate_dehydrogenization).
followed_by(isocitrate_dehydrogenization,
    alpha_ketoglutarate_dehydrogenization).
followed_by(alpha_ketoglutarate_dehydrogenization, succinyl_CoA_synthesis).
followed_by(succinyl_CoA_synthesis, succinate_dehydrogenization).
followed_by(succinate_dehydrogenization, fumarate_hydratization).
followed_by(fumarate_hydratization, malate_dehydrogenization).
followed_by(malate_dehydrogenization, citrate_synthesis).

occurs_in(physical_process,[universe]).
occurs_in(natural_process,[ecosystem]).
occurs_in(live,[organism]).
occurs_in(tca_cycle,[mitochondrial_matrix]).
```

```
                        %%%%%%%%%%%%%%%%%%%%%%%%
                        %%%%% compartment.pl %%%%
                        %%%%%%%%%%%%%%%%%%%%%%%%
:-multifile compartment/2.
:-multifile limits_of/2.

%% limits_of(compartment, [its_limiting_structures]).

limits_of(intracellular_space, [plasma_membrane]).
limits_of(intermembrane_space,[mitochondrial_outer_membrane,
                               mitochondrial_inner_membrane]).
limits_of(mitochondrial_matrix,[mitochondrial_inner_membrane]).


%% compartment(structure,[its_compartmets]).

compartment(cell, [intracellular_space]).
compartment(mitochondrion, [intermembrane_space,mitochondrial_matrix]).


                               %%%%%%%%%%%%%%%%%%%%%%%%
                               %%%%%% neighbor.pl %%%%%%
                               %%%%%%%%%%%%%%%%%%%%%%%%
:-multifile neighbor_compartments/3.
:-multifile neighbor_structures/2.
:-multifile neighbor_regions/3.
:-multifile regions_of/2.

%% neighbor_compartments(CompartmentOut, [structures], CompartmentIn).

neighbor_compartments(extracellular_space,
              [plasma_membrane, cell_wall:plant_cell],
              intracellular_space).
neighbor_compartments(intracellular_space,
      [mitochondrial_outer_membrane],
      intermembrane_space).
neighbor_compartments(intermembrane_space,
      [mitochondrial_inner_membrane],
      mitochondrial_matrix).


%% neighbor_structures(structure,[its_immediate_neighbor_structures]).
```

```
neighbor_structures(cytoplasma,[plasma_membrane,
       nuclear_envelope,
       mitochondrial_outer_membrane,
       er_membrane,
       vesicular_membrane]).
neighbor_structures(mitochondrial_outer_membrane,
   [cytoplasma,
    intermembrane_space_content]).
neighbor_structures(intermembrane_space_content,
   [mitochondrial_outer_membrane,
    mitochondrial_inner_membrane]).
neighbor_structures(mitochondrial_inner_membrane,
   [intermembrane_space_content,
    mitochondrial_matrix_substance]).
neighbor_structures(mitochondrial_matrix_substance,
   [mitochondrial_inner_membrane]).


%% regions_of(structure, [regions_of_structure]).

regions_of(golgi_complex,[cis_side, trans_side]).
regions_of(dictyosome,[golgi_cisterna, golgi_lamella]).
regions_of(mitochondrial_inner_membrane,[intermembrane_space_side,
 hydrophobic_space,
 matrix_side]).

%% neighbor_regions(structure,region,[its_immediate_neighbor_regions]).

neighbor_regions(mitochondrial_inner_membrane, intermembrane_space_side,
 [hydrophobic_space]).
neighbor_regions(mitochondrial_inner_membrane,hydrophobic_space,
 [intermembrane_space_side, matrix_side]).
neighbor_regions(mitochondrial_inner_membrane, matrix_side,
 [hydrophobic_space]).
```

REFERENCES

Addanki, S., R. Cremonini, and J. S. Penberthy (1991). Graphs of models. *Artificial Intelligence 1-3*(51), 145–177.

Barr, A. and E. A. Feigenbaum (Eds.) (1981). *The Handbook of Artificial Intelligence*, Volume 1. Reading, MA: Addison-Wesley.

Barr, A. and E. A. Feigenbaum (Eds.) (1982). *The Handbook of Artificial Intelligence*, Volume 2. Reading, MA: Addison-Wesley.

Buchanan, B. G. and R. G. Smith (1989). Fundamentals of expert systems. In A. Barr, P. R. Cohen, and E. A. Feigenbaum (Eds.), *The Handbook of Artificial Intelligence*, Volume 4, pp. 149–192. Reading, MA: Addison-Wesley.

Bunge, M. (1977). *Ontology I: The Furniture of the World*, Volume 3 of *Treatise on Basic Philosophy*. Dordrecht, Holland: D. Reidel.

Bunge, M. (1983). *Epistemology and Methodology I: Exploring the World*, Volume 5 of *Treatise on Basic Philosophy*. Dordrecht, Holland: D. Reidel.

Burkhardt, H. and B. Smith (Eds.) (1991). *Handbook of Metaphysics and Ontology*. Munich, Germany: Philosophia.

Carson, E. R., C. Cobelli, and L. Finkelstein (1983). *The Mathematical Modeling of Metabolic and Endocrine Systems*. New York, NY: John Wiley.

Cercone, N. and G. McCalla (1987). What is knowledge representation? In N. Cercone and G. McCalla (Eds.), *The Knowledge Frontier. Essays in the Representation of Knowledge*, New York, NY, pp. 1–43. Springer-Verlag.

Cocchiarella, N. B. (1991). Ontology ii: Formal ontology. In H. Burkhardt and B. Smith (Eds.), *Handbook of Metaphysics and Ontology*, Munich, Germany. Philosophia.

Collinson, D. (1987). *Fifty Major Philosophers: A Reference Guide*. London, England: Croom Helm.

Davis, E. (1990). *Representations of Commonsense Knowledge*. Representation and Reasoning. San Mateo, CA: Morgan Kaufmann.

Davis, R. (1993). Retrospective on "diagnostic reasoning based on structure and behavior". *Artificial Intelligence 59*(1-2), 149–157.

Davis, R. and B. G. Buchanan (1985). Meta-level knowledge: Overview and applications. In R. J. Brachman and H. J. Levesque (Eds.), *Readings in Knowledge Representation*, Chapter Procedural Representations and Production Systems, pp. 390–397. Los Altos, CA: Morgan Kaufmann.

Davis, R., H. Shrobe, and P. Szolovits (1993, Spring). What is a knowledge representation? *AI Magazine 14*(1), 17–33.

Dretske, F. I. (1981). *Knowledge & the Flow of Information*. Cambridge, MA: The MIT Press.

Falkenhainer, B. and K. D. Forbus (1991). Compositional modeling: Finding the right model for the job. *Artificial Intelligence 1-3*(51), 95–143.

Gaines, B. G. (1988). An overview of knowledge-acquisition and transfer. In B. R. Gaines and J. H. Boose (Eds.), *Knowledge Acquisition for Knowledge-Based Systems*, San Diego, CA, pp. 3–22. Academic Press.

Gruber, T. R. (1993, January). Toward principles for the design of ontologies used for knwoledge sharing. Technical Report KSL 93-4, Stanford Knowledge Systems Laboratory, Palo Alto, CA.

Hawkins, J. M. and R. Allen (Eds.) (1991). *The Oxford Encyclopedic English Dictionary*. Oxford, England: Oxford University Press.

Hayes, P. J. and K. Ford (Eds.) (1992). *Reasoning Agents in a Dynamic World: The Frame Problem*. Boston, MA: JAI Press.

Houghton Mifflin (1992). *The American Heritage Dictionary of the English Language* (3rd ed.). Boston, MA: Houghton Mifflin.

Jaeger, M. (1993). Circumscription: Completeness reviewed. *Artificial Intelligence 60*(2), 293–301.

Lehrer, K. (1990). *Theory of Knowledge*. Dimensions of Philosophy. Boulder, CO: Westview Press.

Lenat, D. B. (1989, March). Ontological versus knowledge engineering. *IEEE Transactions on Knowledge and Data Engineering 1*(1), 84–88.

Lenat, D. B. and R. V. Guha (1990). *Building Large Knowledge-Based Systems*. Reading, MA: Addison-Wesley.

McCarthy, J. (1980). Circumscription — a form of non-monotonic reasoning. *Artificial Intelligence 13*(1-2), 27–39.

McCarthy, J. (1985). Epistemological problems of artificial intelligence. In R. J. Brachman and H. J. Levesque (Eds.), *Readings in Knowledge Representation*, Chapter The Knowledge Representation Enterprise, pp. 24–30. Los Altos, CA: Morgan Kaufmann.

McCarthy, J. (1993). History of circumscription. *Artificial Intelligence 59*(1-2), 23–26.

McDermott, D. V. (1987). Reasoning, spatial. In S. C. Shapiro (Ed.), *Encyclopedia of Artificial Intelligence*, pp. 863–870. New York, NY: John Wiley & Sons.

Musen, M. A. (1992). Dimensions of knowledge sharing. *Computers and Biomedical Research 25*(5), 435–467.

Newell, A. (1982). The knowledge level. *Artificial Intelligence 18*(1), 87–127.

Popper, K. R. (1982). *The Open Universe*. Postscript to the Logic of Scientific Discovery. Totowa, NJ: Rowman and Littlefield.

Popper, K. R. (1983). *Realism and the Aim of Science*. Postscript to the Logic of Scientific Discovery. Totowa, NJ: Rowman and Littlefield.

Popper, K. R. (1990). Towards an evolutionary theory of knowledge. In *A World of Propensities*. Bristol, England: Thoemmes. (A longer version of the lecture given on June 9th, 1989 before the Alumni of the School at the London School of Economics).

Reiter, R. (1985). On reasoning by default. In R. J. Brachman and H. J. Levesque (Eds.), *Readings in Knowledge Representation*, Chapter Other Approaches, pp. 402–410. Los Altos, CA: Morgan Kaufmann.

Russell, B. (1965). *On the Philosophy of Science*. Indianapolis, IN: The Bobbs-Merrill.

Russell, S. and E. Wefald (1991). Principles of metareasoning. *Artificial Intelligence 49*(1-3), 361–395.

Shannon, C. E. and W. Weaver (1964). *The Mathematical Theory of Communication*. Urbana, IL: University of Illinois Press. Tenth printing of the original work in 1949.

Smith, B. and K. Mulligan (1982). Pieces of a theory. In B. Smith (Ed.), *Parts and Moments*, pp. 15–109. Munich, Germany: Philosophia.

Sowa, J. (1991). Current issuses in semantic networks (in panel). In J. F. Sowa (Ed.), *Principles of Semantic Networks: Explorations in the Representation of Knowledge*, pp. 13–18. San Mateo, CA: Morgan Kaufmann.

Urmson, J. O. and J. Ree (Eds.) (1989). *The Concise Encyclopedia of Western Philosophy and Philosophers*. London, England: Unwin Hyman.

Weld, D. S. (1989). Automated model switching. In *Proceedings Third Workshop on Qualitative Physics*.

Yates, F. E. (1982). Systems analysis of hormone action: Principles and strategies. In R. F. Goldberger and K. Yamamoto (Eds.), *Hormone Action*, Volume 3A of *Biological Regulation and Development*, pp. 25–97. New York, NY: Plenum.