AL-TP-1992-0051

AD-A261 411

DTIC
ELECTE
MAR 0 2 1993
S D

ARMSTRONG LABORATORY

IDEF5 ONTOLOGY DESCRIPTION CAPTURE METHOD: CONCEPTS AND FORMAL FOUNDATIONS

Christopher P. Menzel
Richard J. Mayer

KNOWLEDGE BASED SYSTEMS LABORATORY
DEPARTMENT OF INDUSTRIAL ENGINEERING
TEXAS A&M UNIVERSITY
COLLEGE STATION, TX 77843

Michael K. Painter, Capt, USAF

HUMAN RESOURCES DIRECTORATE
LOGISTICS RESEARCH DIVISION

93-04254

NOVEMBER 1992

INTERIM TECHNICAL PAPER FOR PERIOD JANUARY 1990 - MARCH 1991

Approved for public release; distribution is unlimited.

93 3 1 028

AIR FORCE MATERIEL COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6573

NOTICES

This paper has been reviewed and is approved for publication.

MICHAEL K. PAINTER, Capt, USAF
Program Manager

BERTRAM W. CREAM, Chief
Logistics Research Division

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE November 1992	3. REPORT TYPE AND DATES COVERED Interim - January 1990 to March 1992

4. TITLE AND SUBTITLE IDEF5 Ontology Description Capture Method: Concepts and Formal Foundations	5. FUNDING NUMBERS C - FQ7624 90 [illegible] PE - 63106F PR - 2940 TA - 01 WU - 15
6. AUTHOR(S) Christopher P. Menzel Richard J. Mayer Michael K. Painter	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Knowledge Based Systems Laboratory Department of Industrial Engineering Texas A&M University College Station, TX 77843	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Armstrong Laboratory Human Resources Directorate Logistics Research Division Wright-Patterson AFB, OH 45433-6573	10. SPONSORING/MONITORING AGENCY REPORT NUMBER AL-TP-1992-0051

11. SUPPLEMENTARY NOTES

Armstrong Laboratory Technical Monitor: Capt Michael K. Painter, AL/HRGA, (513) 255-[illegible]

12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited	12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

An ontology can be though of as a structure for representing knowledge about the world as perceived from different perspectives such that those perspectives can be related to one another. Viewed simply as the *study of what there is*, ontology as an activity is at work across the full range of human inquiry prompted by humanity's persistent effort to understand the world in which it has found itself - and which it has helped to shape. In the context of human-designed-and-engineered systems, ontologic inquiry is motivated not by the search for knowledge for its own sake, but by the need to understand, design, engineer, and manage such systems effectively. IDEF5 development has been targeted at filling a methodological gap for reliably extracting ontologies in a manner that closely reflects human conceptualization of the domains in question. The IDEF5 method does so by enabling practitioners to recognize and validate the nature and structure of a given engineering, manufacturing, business, or logistical domain and then store that knowledge in a usable representational medium.

14. SUBJECT TERMS IDEF, information engineering, information systems, integration, knowledge acquisition, method, methodology, ontology, requirements definition, systems engineering	15. NUMBER OF PAGES 41
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

Preface

This paper describes the research accomplished at the Knowledge Based Systems Laboratory of the Department of Industrial Engineering at Texas A&M University. Funding for the Laboratory's research in Integrated Information System Development Methods and Tools has been provided by the Logistics Research Division of the Armstrong Laboratory (AL/HRG), Wright-Patterson Air Force Base, Ohio 45433, under the technical direction of USAF Captain Michael K. Painter, under subcontract through the NASA Research Institute for Computing and Information Systems (RICIS) Program at the University of Houston. The authors wish to acknowledge and extend a special thanks to the IDEF5 design team whose names are listed below:

Dr. Paula S. deWitte

Mike Futrell

Dr. Richard J. Mayer

Dr. Christopher P. Menzel

Michael K. Painter

Summary

This report presents the results of research towards an ontology capture method referred to as IDEF5. Viewed simply as the study of what there is in a domain, ontology is at work across the full range of human inquiry prompted by the persistent effort to understand the world in which it has found itself--and which it has helped to shape. In the context of information management, ontology is the task of extracting the structure of a given engineering, manufacturing, business, or logistical domain and storing it in a usable representational medium. A key to effective integration is a system ontology that can be accessed and modified across domains and which captures common features of the overall system relevant to the goals of the disparate domains. If the focus is on information integration, then the strongest motivation for ontology comes from the need to support data sharing and function interoperability. In the correct architecture, an enterprise ontology base would allow the construction of an integrated environment in which legacy systems appear to be open-architecture, integrated resources. If the focus is on system/software development, then support for the rapid acquisition of reliable systems is perhaps the strongest motivation for ontology. Finally, ontological analysis has been demonstrated to be an effective first step in the construction of robust knowledge-based systems.

An IDEF5 description of an ontology is a computationally tractable representation of what exists in a given domain. IDEF5 provides the means to identify the primary classes, or *kinds*, of objects within the domain by isolating the properties that define the members of those kinds and the characteristic relations that hold between domain objects. IDEF5 allows such representations to be purposely structured in a way that closely reflects human conceptualization of the domains in question. In IDEF5, differing perspectives on the same domain (e.g., varying levels of granularity) and their interrelations are also supported. Finally, IDEF5 supports the identification of complex kinds (*system kinds*) and the properties and relations that characterize members of those kinds.

Part I: Background, Motivation, and Informal Foundations

Any organized system--a business, university, manufacturing plant--can be considered the resultant of three vectors:

(1) the system *ontology*, i.e., the basic entities that populate the system--personnel, equipment, manufacturing systems, etc.;

(2) the *structure* those entities jointly exhibit--the relations they bear to one another; and

(3) the *processes* they undergo--the changes that take place in the organization over time.

An accurate representation of such a system will thus reflect the information within all three vectors. Currently, existing IDEF[1] methods are geared chiefly toward information of the second and third types: IDEF1 (Information Modeling Method) and IDEF1X (Semantic Data Modeling Method) capture primarily structural information; IDEF0 (Function Modeling Method) and IDEF3 (Process Description Capture Method) capture various types of process information. Of course, since both structural information and process information involve objects in a system, there is the capacity for limited ontology representation within the existing methods. However, as noted below, several important kinds of ontological information are not representable in the languages associated with these methods. Furthermore, these methods do not include techniques specifically designed for eliciting and capturing system ontologies. This suggests that there is a need for a separate method. We intend to substantiate this suggestion and begin laying the groundwork for the needed method, IDEF5.

Like other IDEF methods, IDEF5 should be accompanied by: (1) a rigorous, formal foundation for the method and (2) an accompanying software implementation designed for practical information capture and information modeling. The software tool would be

---

[1] The name IDEF originates from the Air Force program for Integrated Computer-Aided Manufacturing (ICAM) from which the first ICAM Definition, or IDEF, methods emerged. It was in recognition of this foundational work, and in support of an overall strategy to provide a family of mutually-supportive methods for enterprise integration, that continued development of IDEF technology was undertaken.

designed for use by domain experts--people attuned to the way a specific system works. The basic question faced by any domain expert, or by a knowledge engineer working with such an expert, is how to describe the things he or she knows about. A good method will reveal the appropriate sorts of general structures that classify the knowledge being sought smoothly and flexibly--the formal foundation--and provide a rich, powerful, user-friendly environment for eliciting that information from the expert, then store and integrate the garnered information efficiently and effectively. In the following sections, we describe the nature of ontology and ontological information, sketch the proposed IDEF5 formal and methodological foundations for capturing that information, and discuss the general proposed features of an IDEF5 software environment.

Philosophical Foundations: The Nature of Ontology

In Western thought, ontology has chiefly been considered an attempt to divide the world at its joints. In a word, it can be called the study of *what there is*. Historically, ontology arose as the major component of the branch of philosophy known as metaphysics, which deals with the nature of reality in general. Metaphysics is perhaps most often associated with questions typically taken to be beyond the reach of physical science, such as the nature of the soul or the mind, the existence of God, or whether or not we have free will.[2] However, there is no necessary connection between ontology and pure, nonempirical, philosophical speculation. Viewed simply as the study of what there is, ontology is an activity at work across the full range of human inquiry prompted by humanity's persistent effort to understand the world in which it exists--and which it has helped to shape.

Natural science, in particular, can be viewed as an example of ontology *par excellence*. Perhaps the chief goal of subatomic physics, for example, is to develop a taxonomy of the most basic kinds of objects that exist within the natural world--electrons, protons, muons, and their fellows. At the other end of the spectrum, astrophysics, among other things, seeks to discover the range of objects that exist in its domain: quasars, black holes, gravity waves, etc. Similarly, the so-called life sciences seek to categorize and describe the various kinds of living organisms that populate the planet. Such examples can be multiplied, of course, from geology to psychology, chemistry to sociolinguistics.

[2] Unfortunately, in contrast to these deep, important--albeit often unresolvable--questions, in the popular consciousness, the term "metaphysics" has also come to be associated with such pseudo-intellectual bilge as astrology, astral projection, occult "science," and similar nonsense.

This sort of inquiry is not limited to the natural sciences, however. The abstract sciences as well--mathematics, in particular--are at least in part an attempt to discover and categorize the domain of abstract objects: prime numbers, transfinite ordinals, Hilbert spaces, continuous nondifferentiable functions, polynomial algorithms, commutative groups, and so on.

The natural and abstract worlds, however, do not exhaust the applicable domains of ontology. There are vast, *human-designed-and-engineered* systems--manufacturing plants, businesses, military bases, etc.--in which the task is just as relevant and just as pressing. Here, though, the ontological enterprise is motivated not so much by the search for knowledge for its own sake (as, ideally, it is in the natural and abstract sciences) but by the need to understand, design, engineer, and manage such systems effectively.

Ontology, then, is a basic research task common to the natural and abstract sciences on the one hand, and common to the information sciences on the other. In the next section, we present the nature of ontological information in greater detail and discuss its application to the information sciences.

Kinds and Instances

Ontology can be understood to involve several subtasks; four are especially worth discussing here: (i) providing an inventory of the kinds of objects that exist within a given domain according to our best sources of information regarding that domain (e.g., a theory or a domain expert); (ii) for each kind of object, providing a description of the properties common to all and only instances of that kind; (iii) characterizing the particular objects that in fact instantiate the kinds within a system; and (iv) providing an inventory of the associations that exist within a given domain between (and within) kinds of objects.

The first two tasks are common in the physical sciences. Thus, in microphysics, for example, one finds the subatomic world grouped into basic kinds--at the grossest level (in the context of subatomic physics!), leptons and quarks, and beneath them the large variety of subkinds of each of those overarching kinds. Along with each kind, are the properties common to all and only members of the kind, including the specific property values of such attributes as mass, charge, spin, and so on which all the members share. Again, in biology, one finds perhaps the foremost example of classifications into kinds and subkinds and characterizations of the distinctive properties associated with each kind.

The third task of ontology becomes more relevant in contexts where we want to be able to characterize specific individual objects and speak specifically of them and their properties. A basic metaphysical distinction is especially useful in this regard, namely, the distinction between *essential* and *accidental* properties. An essential property of an object, S, is a property that S must have to exist. An accidental property of S, by contrast, is a property that S has but does not require. For example, the number 17 has the property of being prime essentially; it could not possibly have been evenly divisible by anything other than 1 and itself. On the other hand, it has the property of being Dr Menzel's favorite natural number accidentally; if Dr Menzel hadn't existed, or if his affections had been directed toward the number 43 instead, it would have lacked this property (and no doubt would have been none the worse for it). Again, human beings are usually thought to have the property of being human essentially--no one could have been a donkey or a stone instead of a human. On the other hand, all of us could have been (and indeed, have been) a different height; thus one's height is an accidental property.

The usual notion of a kind is a class of objects which all share a common *nature* (i.e., a set of properties that belong *essentially* to all and only members of the kind). Consequently, the properties that make a thing a member of a kind are also those that define its nature as an entity. This definition is, for the most part, quite appropriate in the context of natural science and mathematics. For example, the most natural properties for delimiting biological kinds involve having a certain DNA structure; clearly, this will also be an essential property of the animals in question (on the reasonable assumption that no particular animal could have been a member of a different species). Similarly, the most natural properties for delimiting kinds of subatomic particles (e.g., a certain mass, charge, spin, etc.) will be in terms of analogous underlying structural properties that are essential to the instances of those kinds.

As we will argue, though, this definition is too restrictive for use in the context of human-designed systems. However, there is a closely related conception--alluded to briefly in the first paragraph of this section--that is somewhat more flexible and more applicable in the context of information modeling. In this conception, the properties that define a kind are not necessarily essential properties of the members of the kind. Rather, the membership conditions only specify what properties it takes to be an instance of that kind, irrespective of whether or not those properties are essential to the members. Thus, in this broader conception, a kind K is a class of objects consisting of all and only those things that exhibit a certain set of properties, which we can call the *defining properties of K*.

An example will help to show how this conception of a kind is the more useful one in the context of human-designed systems, and will also help to clarify one way in which an ontology might function in the course of information management. Consider the following representation of the basic ontology of a manufacturing cell composed of five entities. Objects enter the cell and encounter a cutter, then a drill, an inspection station, and two cleaners.

KINDS	DEFINING PROPERTIES
A: Cutter	{Has diamond-cutting tool,...}
B: Drill	{Has high speed motor,...}
C: Inspector	{Has high intensity lens light...}
D: Cleaner and Painter	{Has dust filters, high gloss paint,...}
E: Cleaner	{Has liquid cleaners,...}

This example shows the kinds of objects that populate the system, and lists the defining properties of each kind; a representative defining property (or two) is listed for each. Thus, this example provides an abstract representation of the *general* structure that the manufacturing cell must exhibit at any given time. The property **has a diamond-cutting tool** is a defining property of the kind **Cutter**. However, suppose the cutter has the capacity to switch from diamond-cutting tools to carbide. Then, even though **having a diamond-cutting tool** is a *defining* property of the kind **Cutter**, it is nonetheless an *accidental* property of the cutter; it would lack the property if someone were to swap the diamond-cutting tool for a carbide tool.[3] Thus, its role as a defining property of the kind

[3] There are, of course, some significant philosophical issues involved in the the nature of artifacts; some philosophers, for example, argue for the view--known as *mereological essentialism*--that every part of an artifact, or physical object generally, is essential to it, so that if we swap one cutting tool for another in a cutter, the cutter with the replaced tool ceases to be, and a new cutter comes to exist. The puzzle here goes back to Greek times in the guise of the Ship of Theseus: if we bit by bit replace the planks of a ship with new planks, and simultaneously build a new ship, bit by bit, from the old planks, which ship is which? Is the new ship identical to the original ship because it has the same parts? Or is the rebuilt ship identical to the original ship because of the insignificance of each plank individually to the identity of the whole? Thankfully, we needn't address, or at least answer, such questions. The chief purpose of ontology modeling, and information modeling generally, is not so much to divide the world at its ontological joints to discern its ultimate nature, but rather simply to categorize it in the most useful way for the purposes at hand. And the fact is that, in our ordinary ways of thinking about such matters, ordinary objects do not

means only that at any given time, the cutter in the manufacturing cell must have a diamond-cutting tool, irrespective of whether it has a diamond-cutting tool essentially or accidentally.

The general point here is that things can belong *contingently* to important kinds of objects within human-designed systems. The reason for this is that the kinds within such a system are usually *artifacts* (human constructions) and hence an object of one kind might "mutate" into an object of another kind simply by undergoing some nondestructive change (e.g., the exchange of cutting tools). Compare this with an electron which decays into two pions (a destructive change more typical in natural systems); the original object does not survive but is replaced by two distinct objects of a different kind.

In other words, we use the broader notion of a kind because when we build an ontology for a certain human-designed system, we are not necessarily attempting to discover and classify the world as it is in itself, but rather trying to divide and categorize the objects within the system in useful and informative ways. An ontology categorization scheme is justified only insofar as it is useful to organizing, managing, and representing information in the system so categorized. If objects of a certain kind, K, play a useful role in the system, that is sufficient justification to admit them into the system ontology, irrespective of whether the defining properties of K are essential to its members.

The third subtask of ontology is operative in the above example as well. In addition to listing and characterizing the kinds that define the manufacturing cell, we have also discussed the natures of some of their possible instances (e.g., whether they have a certain property essentially and whether that property is a part of their nature). This is no mere philosophical exercise. It might well be crucial in distinguishing the essential from accidental properties. The essential properties of a thing, S, put inviolable *bounds* on what is possible within a system containing S. If S has the property P essentially, it cannot fail to have it. For example, a design that specifies a kind that includes a property that precludes P among its defining properties cannot use S as one of its instances, regardless of how well it might meet the remaining specifications.

---

cease to exist if we change relatively insignificant parts. As a matter of fact, our theory will remain neutral on this question and will permit, though not require, mereological essentialism should it prove useful in some contexts, as it conceivably might.

There is more to characterizing the objects in a system than listing their properties. In the context of a given system, it is equally important to detail the *relations* that objects in the system can, and do, bear to one another. Considerations such as those above lead us to distinguish *system-essential* from *system-accidental* relations. A system-essential relation relative to two (or more) kinds (K_1 and K_2) is a relation that must hold whenever there are instances of K_1 and K_2. A system-accidental relation relative to K_1 and K_2, by contrast, is one that need not hold between all possible instances of those kinds. For example, the nature of the manufacturing cell depicted above might require a certain sort of informational link to be established between the cutter and the drill which informs the drill of the type of operation the cutter has performed on a given piece of material. In ontological terms, this would then be characterized as a system-essential relation relative to the kinds **Cutter** and **Drill**. On the other hand, the spatial relationship between cutter and drill may well be irrelevant (e.g., the drill might just as well be three feet north of its actual location in the cell). In this case, we say that the *de facto* spatial relationship between cutter and drill is system-accidental.[4]

An interesting example of a system-essential relation is the **part-of** relation that often holds between a complex object and some of its parts. Consider an engine of a specific design. The engine can itself be viewed as a complex system, made up of many smaller parts. Each of these parts can be classified as instances of a kind, as can the engine itself. Given some kind of part, **P**, that is necessary to the design of the engine, **E**, then, relative to **P** and **E**, the **part-of** relation is system-essential. However, given an instance **e** of **E** and the instance **p** of **P** within **e**, some other instance **p*** of **P** would have done just as well. Hence, the **part-of** relation does not hold essentially between the instances **p** and **e**.

As this example shows, entire systems can themselves be considered further objects in yet larger systems and can be characterized as possessing certain properties (e.g., a manufacturing cell which comprises five machines). This means that an adequate ontology tool will have the capability to examine and characterize the system from the coarsest to finest levels of detail.

---

[4]Certain facts about the configuration of the drill or cutter *could*, however, require that the two be oriented in one and only one way. In this case, the relation would be system-essential. Note that, just as defining properties of kinds needn't be essential to their instances, in the same way entities that stand in system-essential relations don't necessarily stand in those relations essentially; though being spatially oriented in a certain way might be essential within the system, the drill and the cutter don't necessarily have to stand in that relation in *any* possible system in which they might exist.

Accumulation of Domain Ontologies

What, exactly, is ontology good for? What role can it play in the design and development of information systems? In what sorts of information modeling contexts will it be useful?

One of the most important aspects of the general development and use of the IDEF5 method will be the accumulation of a wide range of domain ontologies. Generally, one of the greatest problems in information management is inefficiency. Redundant effort is expended to capture or re-create information that has already been recorded elsewhere. Consider the analogy with programming. Very often the same kinds of routines (e.g., in the design of user interfaces) are used repeatedly in different programs, typically by different programmers. Enormous amounts of time and effort have thus gone into reinventing the wheel over and over again. Recognition of this problem has led to the development of vast *libraries* that have been collected over time which contain oft-used routines which a programmer can simply call into his or her program, rather than having to duplicate the function of existing code.

Information management across similar settings faces the same sort of problem. Manufacturing domains, for example, share many common features; the more similar the domains, the more features they share. Rather than encoding this information again for each new setting, we propose to develop an analogue of the concept of a programming library by collecting this common information into *ontology libraries*; that is, large, revisable databases of structured, domain-specific ontological information which can be used in the IDEF5 environment. We envision numerous advantages to such libraries, two of which are especially notable. First, domain experts developing an IDEF5 ontology for a specific system will be able to import relevant portions of the general ontology database for the type of system they are describing directly into their IDEF5s. This will save them the trouble of having to record the information directly, thus providing the analogue to the concept of a programming library. This information will, of course, be malleable so that a given expert can modify it in response to unique features of his or her system. Second, the information can be used to construct general techniques for aiding the domain expert in extracting domain knowledge. For example, by isolating and analyzing general patterns or features of ontologies within certain domains, one can develop productive strategies for eliciting and structuring the sorts of knowledge one is likely to find in those domains. If a particular common type of machine varies in certain details from location to location, the background ontology database can import the common information directly, and lead the user through a series of questions to elicit the specifications unique to his/her domain. Again, an expert may not *know* how a certain object should be classified. By searching a list of essential properties of the object, the tool could return a set of kinds in which the object would most naturally be included.

With an array of ontology databases in use across a wide variety of engineering, manufacturing, business, and logistical systems, the task of information modeling could be revolutionized. Of course, the construction of such databases is an enormous--though we believe quite realizable--task. However, there is an even more basic task. Before one can build any complex physical objects (e.g., a bridge) there must be an appropriate methodological and theoretical foundation. This is no less true for abstract objects like information models. That is, before we can think about the structure of a domain-specific ontology database, we need formal theoretical foundations for ontology proper (e.g., the appropriate representational medium) and methodological foundations for the capture and storage of ontological information.

Ontology and Existing Methods

The goal of IDEF5 is not to define yet another method to do something a little better or a little different than some other existing method. We have no interest in and see no virtue to instigating another skirmish in the methodology wars. Rather, our first goal is to point out a *gap* in the existing set of methods: there is, we believe, a type of information--ontological information--that has not been directly targeted by any existing method. Our second goal is to make some preliminary suggestions for filling that gap, both theoretically and practically.

Thus far, we have outlined the nature of ontological information. The importance of this sort of information should be clear. What is perhaps less clear is the need for a new method to capture this information. That issue will be addressed in this section.

For those familiar with other IDEF methods, the idea of capturing information about kinds and their associated properties will no doubt suggest both IDEF1 and IDEF1X. A kind has been defined above as a certain sort of class; this might suggest that a kind is like an IDEF1 *entity class* or an IDEF1X *entity*. Furthermore, associated with each entity class (entity) is a list of associated *attributes* that assign property values to the members of the entity class. Perhaps the makings of an ontology modeling method are in one of these two methods.

We will begin by addressing IDEF1. It would be a serious error to think of IDEF1 as an ontology modeling tool because ontology modeling is *real-world* modeling. That is, the members of kinds are real-world objects, the actual instances of those kinds that exist within the system being modeled. The members of an IDEF1 entity class, by contrast, are *information objects*--they are objectified clusters of information about a system that must be kept, the various "information images" of the real-world objects within a system. Such objects are *defined by* the information they encode. Thus, all the property values associated with an IDEF1 information object are essential to that object; altering a value results in a new object.

This view has two consequences relevant to ontology. First, there will generally not be a one-to-one correspondence between the information objects within an IDEF1 model and the real-world objects being modeled. For instance, within an IDEF1 model of a certain business there might be an entity class, MANAGER, and another entity class, EMPLOYEE. These will be different entity classes since they keep different kinds of information. Thus, an employee who is also a manager would generate two distinct information objects (one for each class): one for the employee in his/her role as an employee, and another for that same real-world employee in his/her role as a manager. Thus, it would be confusing to think of IDEF1 as an ontology modeler; it is simply not designed to represent that kind of information. Second, since all the properties of an information object are essential to it, there is no room for the distinction between essential and accidental properties.

To press the issue further, suppose we overlook the above problems and use IDEF1 as an ontology modeling tool to represent kinds as entity classes. Here then is another difficulty. Suppose some members of a certain kind of engine widget have an additional, removable part--a FRAMMITZ--that, depending on its location on the widget, makes them suitable or not for use in engines of various sizes. Then, having a frammitz and its location on a given widget are accidental properties associated with the kind WIDGET; members can either have them or lack them. Members that have them can come to lack them but, nonetheless, they are important properties to be aware of and track. The inapplicability of accidental properties in IDEF1 has already been noted. However, in IDEF1, the No Null Rule states that every attribute associated with a given entity class must yield a corresponding value for every member of the entity class. Thus, returning to the example, *location_of_frammitz* cannot be a legitimate attribute in an IDEF1 representation of the kind WIDGET, since not every widget has a frammitz (i.e., the value of *location_of_frammitz* for some widgets is null).

In IDEF1, one can capture the information in question without violating the No Null Rule by inventing a new class of entity--WIDGET_WITH_FRAMMITZ. However, there are several problems with this in the context of ontology. First, a matter of ontological aesthetics and to paraphrase Ockham's Razor, one should not be forced to multiply entity classes beyond necessity nor to represent the information in question by introducing an entirely new entity class. However, and more importantly, despite the significant degree of freedom allowed in constructing an ontology for a human-designed system, one is still constrained to make natural and useful divisions into kinds. But a class like WIDGET_WITH_FRAMMITZ does not represent such a division. From the perspective of ontology, it is an artifact foisted upon the modeler by the given modeling tool. The information in question is more accurately and appropriately captured by identifying the class of frammitz-bearing widgets as a mere subclass of widgets with members which belong to the class contingently than by identifying a separate, overlapping kind.

One might suppose, then, that we will fare better with IDEF1X. Although there is some disagreement about the exact semantics of IDEF1X diagrams, it is clear that the members of an IDEF1X "entity" (the spectacularly ill-advised IDEF1X term for a class of similar objects in a system) are to be considered as real-world objects, not information images of those objects (as in IDEF1). Thus, an IDEF1X model in the **EMPLOYEE/MANAGER** example would be thought to contain the same real-world object in both the **EMPLOYEE** and **MANAGER** entities. In an ontology model of the same example, the kind **EMPLOYEE** and subkind **MANAGER** would be considered in the same way. Furthermore, with its capacity for expressing the subclass relation, the recommended analysis of the **WIDGET** example in the previous paragraph could be expressed in IDEF1X. Perhaps IDEF1X is the only model needed.

However, there are deeper limitations. Chief among these is that IDEF1 and IDEF1X are purposely designed with certain expressive limitations which constrain the structure of the information they represent. This makes for very clear, uncluttered, and efficient information and data models. However, it also limits the applicability of IDEF1 and IDEF1X outside their intended domains. The inability of IDEF1 to distinguish essential from accidental properties was illustrated above. The same problem is shared by IDEF1X, as illustrated in the manufacturing cell example above. Suppose, for security reasons, we want to make it impossible to swap out the diamond-cutter tool in the cutter. That is, suppose we want to specify in the list of defining properties of the kind **Cutter** that any instance must have a diamond tool *essentially*. Without the capacity to express modal information, this is not possible; in particular, it is not possible to express this in IDEF1X. Nonetheless, as the example illustrates, it may be of singular importance to be able to express such information.

Further examples abound. For instance, in both IDEF1 and IDEF1X it is not possible to name individual objects in an ontology and assert things specifically about them. Rather, one can only say things that hold of *every* member of a given class of entities in general. This is a crucial limitation in cases where there is a distinguished member of a given kind with special properties. Also, more germane to the current context, it effectively rules out the possibility of carrying out the third task of ontology. If one cannot say *anything* about specific objects, he/she cannot discuss the properties they have. Again, the two methods can express only a limited variety of general propositions about the structure of the entities within a given class. For instance, one might want to note that for every member of class A with property P, there is another member with property Q. This is a straightforward quantificational statement, easily expressed, for example, in predicate logic. Once again, this proposition is beyond the expressive capabilities of IDEF1 and IDEF1X. However, as shown in the previous examples, this might well need to be expressed to provide a thorough characterization of the objects in a system.

The overarching point is that the existing IDEF methods were simply not designed for ontology modeling; they were designed to meet other goals. Again, the claim is not that there is something wrong with or inadequate about the existing IDEFs. They were simply not designed to be tools for ontology modeling and should not be expected to meet the requirements of such a tool.

Increasing Expressive Power

First and foremost among the requirements of an ontology tool is greater expressive power. This need will be met in the theoretical foundations of IDEF5 by imbuing its underlying formal knowledge representation language with the full power of first-order modal logic. The power of first-order logic is well-known and greatly exceeds the expressive power of IDEF1.[5] Modal logic extends first-order logic by introducing modal operators for necessity and possibility and a corresponding set theoretic semantics. This extension, among other things, gives one the power to express facts about essential and accidental properties in a very natural way. Recall that an essential property of x is one which x must have to exist.

The standard set-theoretic semantics for modal logic is discussed in terms of the heuristic concept of a "possible world." The idea goes back to the philosopher/mathematician Leibniz. Most of us believe that there are many ways the world could be other than the way it is. These *ways the world could be* can be thought of as other possible worlds. One way the world could be, of course, is the way the world actually is. Thus, the *actual world* is one of the possible worlds. Unlike possible worlds, though, it is actual, not merely possible. An object S is said to *exist in* a possible world, W, just in case S would have existed if W had been actual. An essential property of an object S is a property that S could not have lacked. In the possible worlds picture, this can be defined as: property p is essential to S just in case S has p in every possible world in which S exists. Correspondingly, p is accidental to S if there is some world in which S exists and fails to have p.

It is often illuminating to think of systems in terms of possible worlds. In importing the enterprise of ontology into the information modeling domain, we noted that our concern was not with the world *per se*, but rather with the world of an organized system. Accordingly, in this context, possible worlds should not be considered alternative states of the world but as alternative states of the system. Thus, a relational database model could be thought of as modeling, in one fell swoop, all the possible states of the database being modeled (i.e., all the different possible relations that could populate the database).

---

[5] Nonmodal first-order logic is developed and discussed in some detail in "Theoretical Foundations for Information Representation and Constraint Specification," Armstrong Laboratory Report AL-TP-1991-0044, October 1991.

Thinking in these terms often helps one to design more breadth and flexibility into the model in anticipation of possible but unlikely or previously unconsidered states. In addition to providing a definition of essential and accidental properties, the possible worlds picture helps anticipate or consider all possible natural kinds that might appear within the system; thus, it defines a sufficiently broad ontology.

A caveat is necessary here to head off a potential misconception. The intuitive concept of a possible *world* might suggest the idea of *completeness* or *totality*; a world, after all, is a total system, complete in every detail. However, the use of worlds in our formal apparatus might suggest that, for us to have an acceptable model of a given system, we must capture every piece of information within the system down to its last detail. But then informationally incomplete models like the simple manufacturing ontology model, **M**, above will not do; we will have to fill in all the informational details before we have an acceptable model. For example, in a system represented by **M**, each machine consists of parts not mentioned explicitly in the model. Also, each part meets certain specifications that were not mentioned, has a certain origin (e.g., a particular vendor) that was not mentioned, and so on. Practically speaking, this descending chain of information is unending. Similarly, any two objects within the system can in principle be regarded as a further object. There is often a call for such representations. For example, in a system, **M**, the cutter and drill may be integrated in such a way that it is useful to regard them jointly as a single object; yet, no such object is represented in **M**. Hence, the notion of a world seems to put far too great a demand on the modeling enterprise.

Fortunately, this is not a genuine problem. The notion of a world should not be taken too literally. Formally speaking, worlds are just indexed structures that (in a modeling context) represent possible or successive states of a system. These structures themselves can be as sparsely or as richly detailed as the modeler desires, depending on how much detail he or she wishes to capture. In particular, a formalized version of the model **M**, with just that much detail, would be a fully acceptable "world." Since there is no finite upper bound to the amount of detail that can be stored within this framework, one can add detail or new objects in whatever fashion is deemed appropriate.

The efficacy of the framework of possible worlds is witnessed by the fact that it is more or less the framework chosen by the members of the International Organization for Standardization (ISO) working group for characterizing the notion of a conceptual schema.

A conceptual schema consists of all the *necessary* propositions that hold in a given system, all possible worlds, or all possible states of the system. Thus, our use of the framework ties in naturally with our work on the development of the three-schema architecture.

Part II: Methodological Foundations

Our methodological experience in ontology development is based on practical industrial applications with Chrysler, Sematech, and our work on the emerging Air Force IDEF5 ontology description capture method. IDEF5 encapsulates the best practice experience in ontology development of the information management community at large to date. The work with Sematech took place in the manufacturing and engineering domain; the work with Chrysler occurred in the product design domain. The experiences at both companies in developing ontologies was found to be remarkably similar. The still-formative methodology sketched below is based on this experience Broadly stated, the procedure consists of the following five steps (brief annotations follow the statement of each step).

Step 1 - Scope domain and collect raw data. This task is responsible for: 1) determining the boundaries of a domain, 2) performing interviews with the domain experts, and 3) collecting samples of data representative of the inputs, controls, policies, knowledge, and products of the domain.

Step 2 - Develop initial proto-kinds. This task is responsible for the analysis of raw data to generate a tentative *relation-poor* ontology of *proto*-kinds, proto-situations, and proto-situation types. By a *relation-poor* ontology, we mean that system-essential relations of kinds are not yet considered in detail at this point (see the annotation to Step 4 below). By a *proto*-kind (-situation, -situation type) we mean a tentative kind (situation, situation type) generated from observation and/or a cursory analysis of existing sources of information. This provides a very useful, albeit defeasible, *rough draft* ontology to guide further inquiry and analysis.

Step 3 - Refine initial analysis. This task is responsible for validating initial proto-kinds and generating a more stable (but still relation-poor) ontology from the tentative ontology. Further inquiry and analysis guided by the tentative ontology gradually yields a revised and more stable ontology. Stability is of course a relative notion. Our experience confirms, however, that careful analysis can come close to the ideal.

Step 4 - Add relations. This task is focused explicitly on adding system-essential relations to the ontology. The chief reason for focusing on system-essential relations is to prevent an unwarranted explosion of complexity. If a significant number of relations are introduced into the tentative ontology, it can become extremely messy to untangle, reassess, and refine the initial relational connections. Furthermore, adding relations early can be misleading, since the ostensible occurrence of a relation involving a nongenuine kind can prejudice a modeler's assessment of the reality of that kind.

Step 5 - Validate stable ontology using raw data. This task is responsible for validating the stable ontology by taking the initial raw data and attempting to *instantiate* it, i.e., model it within the stable ontology. Where this does not prove possible, or where it proves inordinately awkward, the ontology is modified appropriately.

At each step in the process, the results will be distributed for peer review and comment. In our experience, Steps 1, 3, and 4 were found to work very well in team contexts. Step 2, the move from a tentative ontology to a stable one, appears to be most effectively done by an individual. As opposed to team-oriented steps, fairly refined attunement to certain patterns within the system seem to appear when an individual works alone to develop the tentative ontology from the raw data.

Part III: The IDEF5 Description Development Environment

Levels of Data Entry

First-order logic is powerful and efficient; however, a good bit of experience is required to master the art of translating ordinary language into it. Thus, we envision an environment that will permit several levels of data entry. Those familiar with logic should be able to enter information in that format directly. A level up from direct entry will be the possibility of graphical entry. Several graphical representations of first-order logic have been developed; many explicitly for the end-of-knowledge representation.[6] We will be drawing on this work to develop a graphical representation of first-order modal logic. The modal component in particular will require work beyond what is currently available.[7]

In conjunction with the graphical language, we will incorporate the capability for guided, structured text entry. The form of such entries will be midway between straight first-order modal logic and unconstrained natural language. We are fully cognizant of the severe, perhaps intractable, difficulties of full natural language processing (NLP) and do not expect that we will be able to develop a full-blown NLP component for the IDEF5 environment (though it will certainly be capable of incorporating the current state-of-the-art). However, our experience and that of others in developing *constrained* natural language environments has shown that users, with relative ease, learn to express their thoughts within certain syntactic guidelines.[8] Developing such guidelines in the IDEF5 environment will then permit entry of data in a manner that is relatively natural and easy to learn, but which is either immediately processable by the software or easily converted into processable form. The facility will include on-line guidance for proper entry and an appropriate amount of built-in syntax checking to assist the user without confusing or defeating him or her.

---

[6] See [Sowa 84] and [Burch 91] for two representative examples.

[7] One of the research team's stronger areas of expertise is in the area of modal logic. Cf. C. Menzel, "The True Modal Logic," forthcoming in the *Journal of Philosophical Logic*; also C. Menzel, "Actualism, Ontological Commitment, and Possible World Semantics," forthcoming in *Synthese*.

[8] Cf. P. Mayer, "A Computational Approach for Processing Locative and Temporal Information in Clinical Medical Records," unpublished Ph.D. dissertation, Department of Computer Science, Texas A&M University, 1989; also P. Mayer et al., "Locative Inferences in Medical Texts," *Journal of Medical Systems* **11**, 68-85, (1987).

Finally, the IDEF5 environment will also allow straight text entry for those unfamiliar with the graphical or first-order languages, and quick collection of domain knowledge that can be analyzed more formally at a later time.

Hooks to Other Methodologies

Our chief goal in developing and extending the suite of methods is *data integration*. Thus, we envision the IDEF5 environment itself to be smoothly integrated with the other IDEF method support tools, as well as with tools developed for other, related methods such as ER[9] and NIAM.[10] Our efforts are thus geared toward the development of a comprehensive information modeling/knowledge engineering environment capable of storing, integrating, and reasoning with information across various types of domains.

---

[9]Entity-Relationship (ER) modeling method.
[10]Nijssen Information Analysis Method (NIAM).

Part IV: Formal Foundations

In this section we provide a formal language and model theory for ontology, and indicate the roles of the various elements of the formalization (i.e., to which aspects in the above informal development they correspond).

Model Theory

We begin with the notion of a *basic ontology model structure (boms)*, which is a representation of a system ontology (at some level of development and detail). More precisely, a boms, **M**, is an 8-tuple ·D,W,@,d,£,K,R,pÒ in which D and W are mutually disjoint nonempty sets, @ŒW, d : W Æ Pow(D) (i.e., d is a function from W into the power set--set of all subsets--of D). Intuitively, D is the set of all possible individuals; W is the set of all possible worlds or, more relevantly, all possible states of a given system; and @ is the *actual* world, or actual system state. Then, d must be considered a function that assigns to every possible world, wŒW, the subset of D that consists of the possible individuals that exist in w. Consequently, d(w) is called the *domain* of w.

The last four elements of **M** need a little more discussion. First, for all natural numbers, *n*, let F_n be {f | f : W Æ Pow(D^n)}; i.e., the set of all functions from W into the set of all sets of *n*-tuples of elements of D. F_n is the standard possible world semantical definition of the set of all *n*-place *relations*; in particular, F_1 is the definition of the set of all *properties*. The idea behind this definition is that, whatever properties ultimately exist, it is intuitively clear that corresponding to each property in any given world is the set of all the things in that world which have the property. For example, corresponding to the property **redness** in the actual world is the set of all things that are actually red; in another world, there is a different set. This suggests that, rather than seeking any deeper analysis, we simply *identify* **redness** with these varying sets, or more precisely, that we identify it with a *function* that, in each world w, picks out exactly the red things in w. To have the property **redness** in a given world w is thus simply to be in the set of things (the red things, of course) picked out by the property in w.[11]

[11] This is the standard "possible worlds" definition of properties and relations. The account has suffered much criticism from philosophers and linguists of late because it is *coarse-grained*, i.e., properties and relations that pick out the same sets in all possible worlds are identical. However, intuitively, the objection goes, properties and relations can be necessarily coextensive without being identical, e.g., the properties **triangularity** and **trilaterality**. Though important, it is our belief that these issues do not

In this account of properties and relations, the function d in **M** which assigns a domain of objects to each world, $w \in W$, is a property, namely the property **existence**. It is a function which assigns to each world, w, the set of objects that exist in w. The distinction between essential and accidental properties is captured straightforwardly in this framework. As noted above, intuitively, an object has a property, p, essentially (just in case it has it in every possible world in which it exists) and p accidentally (just in case there is some world in which it exists but lacks p). This translates as follows: an object a has the property $p \in F_1$ essentially for all $w \in W$ such that $a \in d(w)$ (i.e., for all worlds in which a "exists"), $a \in p(w)$); and for relations generally, objects $a_1, \ldots, a_n$ stand in the relation $r \in F_n$ essentially for all w such that $a_1, \ldots, a_n \in d(w)$, $\langle a_1, \ldots, a_n \rangle \in r(w)$. Similarly, a has the p accidentally in case $w \in W$ such that $a \in d(w)$ but $a \notin p(w)$.

Given the definition of F_n, we can specify the character of the remaining elements of **M**. First, we stipulate that $\leq \in F_2$--i.e., that $\leq$ is a two-place relation on possible individuals--and that for each $w \in W$, $\leq(w)$ is a *reflexive partial ordering* on the domain d(w) of w. That is, writing $a \leq_w b$ for $\langle a,b \rangle \in \leq(w)$, for all $a \in d(w)$, $a \leq_w a$ (reflexivity), and for all $a,b,c \in d(w)$, if $a \leq_w b$ and $b \leq_w c$ then $a \leq_w c$ (transitivity). Intuitively, $\leq$ represents the *part-whole* relation; thus, for all $w \in W$, $\leq(w)$ is the set of pairs $\langle a,b \rangle \in d(w)$ such that a is a part of b in the world or system state w. Thus $a \leq_w b$ can be read as a *is a part of b in w*. We write $a <_w b$ if $a \leq_w b$ and $a \neq b$, and say that a *is a proper part of b in w* if $a <_w b$. We also say that a is *simple in, or relative to*, w if a has no proper parts in w. If a is not simple in w, then we say that a is *complex*, or *a system, in w*.[12]

As stressed above, part-whole relations are crucial for the accurate representation of physical systems, especially manufacturing and engineering systems, and this additional structure imposed on the objects of each possible world (possible system state) captures those relations in a simple but powerful way. Note that since the relation is partial, it can

---

typically affect ontology or information modeling generally, and hence the complexities of finer-grained accounts can be avoided. Cf. e.g., J. Barwise and J. Perry, *Situations and Attitudes* (Cambridge, MIT Press/Bradford Books, 1983), ch. 2; G. Bealer, *Quality and Concept* (Oxford, Oxford University Press, 1980), ch. 2.

12The idea of adding additional algebraic structure on each world's domain of individuals to capture the part-whole relation was inspired by the work of Godehard Link on the semantics of plurals. Link imposes a full-blown boolean algebra on the individuals to provide interpretations for a wide variety of plural phenomena in natural language, and this seems to be far more structure than is necessary for present purposes here. Link also restricts his attention to nonmodal contexts. See G. Link, "The Logical Analysis of Plurals and Mass Terms: A Lattice Theoretic Approach," in R. Bauerle et al. (eds), Meaning, Use, and Interpretation (Berlin, De Gruyter, 1983).

be as elaborate or as sparse as required; everything from the empty relation to a linear well-ordering counts as a partial ordering. The requirements of reflexivity and transitivity guarantee only that every object is a part of itself, and that the parts of the parts of an object *a* are also parts of *a*. In particular, because models need not be *complete* descriptive representations, the part-whole relations between objects in a model can be as detailed or as sparse as desired. This makes for great flexibility in the development of models, since it allows one to add part-whole information incrementally in the construction of a model to whatever extent is deemed necessary. Note also that the part-whole relation needn't hold essentially between two objects. That is, it is perfectly consistent within a model for *a* to be a part of *b* in one world w and for *a* not to be a part of *b* in another. This implements the idea discussed above (see footnote 2) that, intuitively, most complex objects do not have all of their parts essentially.[13]

The sixth element, K O F_1, is a set of properties that represent the kinds within a system; hence, the members of K are called the *M-kinds* or *kinds of M*. In our informal development above, kinds were identified with *classes*, which are usually considered collections of some kind. However, kinds cannot be thought of as mere collections because they transcend their members; the nature of a kind is not altered if its instances change. This is precisely the feature of properties noted above that distinguishes them from sets. Thus, in our more precise development, kinds are best identified with certain distinguished properties, and hence K is stipulated to be a subset of the set of properties F_1.

The seventh element, R, of **M** represents the system-essential relations; hence, we stipulate that R O »$_{n\geq 2}F_n$, i.e., that R is a subset of the set of all 2-or-more-place relations. The final element, p, is defined as a function of K»R such that p : K Æ Pow(F_1-K), and p : R Æ »$_{n\geq 2}K_n$ such that for all *n*-place relations rŒR ($n\geq 2$), p(r)ŒK_n. Therefore, the role of p, intuitively, is to map each kind, k, to the set of its defining properties of k, and to map each relation, r, in R to the kinds relative to which r is system-essential in w. That no kind is the defining property of some other kind (or itself, for that matter) is ensured by the stipulation that p maps K *into* Pow(F_1-K), rather than Pow(F_1) simpliciter. The stipulation that p be one-to-one assures that no two distinct kinds have precisely the same defining

[13]Note, however, that having a filter might be a defining property of the kind **cleaner**, so that in any state w of a given system, all cleaners must have filters. We have dwelled on this point already above, but if an ontology is not to be muddled, it is crucial that the distinction be clear.

properties. Note that the defining properties of one kind, k, might constitute a (proper) subset of the defining properties of another kind, k¢, so that every instance of k¢ is an instance of k. In such a case, we say that k¢ is a *subkind* of k. For example, one might wish to define a general kind, **cutter**, and two separate subkinds, **diamond-tool cutter** and **carbide-tool cutter**, which were obtained by adding additional properties to the more inclusive kind. By defining the defining properties of a kind independent of any world, we build in the idea that they are essential to it. One's *conception* of a particular kind might change over time, of course, but this can be represented in terms of a series of several distinct but related kinds.

Further stipulations about the behavior of p must be made in order to assure that defining properties and system-essential relations are represented correctly in **M**. Specifically, we add two conditions on p. First, if kŒK, then for all pŒp(k), k(w) O p(w), for all wŒW, i.e., in any world w, every member of the kind k in w must have the property p. Second, in the same manner, for any *n*-place relation rŒR such that p(r) = ·k_1, ..., k_nÒ, for any wŒW, if k_i(w)πΔ for all *i* such that 1£*i*£*n*, then there are a_1, ..., a_nŒd(w), a_iŒk_i(w), 1£*i*£*n*, such that ·a_1, ..., a_nÒŒr(w). This condition captures the system-essentiality of system-essential relations. Specifically, the condition states that for any world, w, whenever each of the kinds relative to which r is system-essential has at least one member in the domain of w, then r holds between members of those kinds in w.

An important relation that can occur between models is that one can be *embedded* in another in the sense that all the information in one model is preserved in another model which contains more information. If a model, **M**, is so embedded in another ,**M¢**, we say that **M** is a *submodel* of **M¢**. This sort of situation can arise in at least two ways. First, it is an essential fact of the modeling enterprise that models evolve over time. One of the circumstances under which this happens is when an existing model must be augmented in ht of new information. Another is when one might purposely filter information to obtain a simpler, more coarsely grained model--after all, not all available information is useful in all contexts; thus, one might freely filter the information in a given comprehensive model in a variety of ways to obtain many different submodels.

To begin, assume that **M** = ·D,W,@,d,£,K,R,pÒ is a *substructure* of **M¢** = ·D¢,W¢,@¢,d¢,£¢,K¢,R¢,p¢Ò if and only if D O D¢, W O W¢, @ = @¢, and d(w) = d¢(w)«D, for all wŒW. Suppose **M** is a substructure of **M¢**, let rŒ»$_n$$F_n$ be an *n*-place relation of **M**, and let r¢ be an *n*-place relation of **M¢**. Then we say that r is the *restriction* of r¢ to **M**, written r¢≠**M**, just in case, for all wŒW, r(w) = r¢(w)«D. **M** is a *submodel* of **M¢** just in case **M** is a substructure of **M¢**; £ = £¢≠**M**; for each kŒK there is a k¢ŒK¢ such that k = k¢≠**M** (such a k¢ is called a *correlate of k in M ¢*) and similarly, for R and R¢; for each kŒK, and for each pŒp(k), there is some p¢Œp¢(k¢) such that p = p¢≠**M**, where k¢ is a correlate of k in **M¢**; and for each *n*-place rŒR, p(r) = ·k¢1, ..., k¢nÒ, where k¢i, 1£*i*£*n*, is a correlate of k in **M¢**.

That is, **M** is a substructure of **M¢** if the individuals and worlds of **M¢** include those of **M** they share the same actual world, and the individuals that exist in a world of **M** are exactly those individuals of **M** that inhabit that world in **M¢**. Thus, all the individuals that inhabit that world according to **M** also inhabit it according to **M¢**, though **M¢** may include new individuals in that world as well. The remaining conditions that must be met in order for **M** to be a full-blown submodel of **M¢** simply spell out the idea that the properties and relations of **M**--in particular, the part-whole relations, the kinds, and the system-essential relations of **M**--can only change in **M¢** in ways that increase information, i.e., such that none of the information of **M** is lost. For example, if *a* is part of *b* in w relative to **M**, then *a* is a part of *b* in w relative to **M¢**--though there may be some part, *c*, of *a* in w relative to **M¢** that was not recognized in **M** because *c* is not among the individuals in M. This corresponds to a situation in which **M¢** represents a finer-grained representation of a system also represented by **M**. Again, a kind, k, may have more defining properties in **M¢** than it had in **M**, but those in **M¢** that have correlates in **M** will still be true of all the objects in **M¢** that they were true of in **M** (plus perhaps some that were not among the individuals of **M**). It may be, however, that certain defining properties of k in **M¢** were not recognized in **M** because they only appear at a finer level of granularity, or because of some other shift in perspective not captured by **M**.

Languages for Ontology

In this section, we present the formal language for ontology and discuss the development of more user-friendly, graphical languages for use in the IDEF5 description development environment.

As noted, the formal IDEF5 language L will be a modal extension of first-order logic. It will thus consist of:

- a possibly infinite store of individual constants c_1, c_2, ...,

- individual variables v_1, v_2, ...,

- n-place first-order predicate constants P^n_1, P^n_2, ..., and

- n-place predicate variables F^n_1, F^n_2, ...,

for any or all n as desired, though it is required that L at least contain all variables and the predicates P^2_1, which will ordinarily be written as =, as well as the predicate P^2_2, which will be written as e. e will express the part-whole relation in L. In addition, L will contain the standard logical operators \$ (existential quantifier), ¬ (negation), and & (conjunction), as well as the modal operator ‡ (possibility). L differs from typical first-order modal languages because it contains predicate variables as well as distinguished higher-order predicates KIND, DP, and SER_n ($n \geq 2$) which express the property of being a kind, the relation between a kind and its defining properties, and the relation between a system-essential relation and the kinds relative to which it is such, respectively.

The syntax of the formal language will also be standard, modulo the special higher-order predicates.

- If P is an n-place first-order predicate (constant or variable) and t_1, ..., t_n any n terms (i.e., constants or variables), then $Pt_1, ..., t_n$ is a (first-order atomic) *formula* (of L).

- If P is a one-place first-order predicate, then KIND(P) is a (second-order atomic) formula.

- If P and Q are one-place first-order predicates, then DP(P,Q) is a (second-order atomic) formula.

- If P is a two-place first-order predicate and P_1, ..., P_n ($n \geq 2$) are one-place first-order predicates, then SER_n $(P, P_1, ..., P_n)$ is a (second-order atomic) formula.

- If j and y are formulas, so are ¬f, ‡f, and (f & y).

- If j is a formula and c is any variable (individual or predicate), then $aj are formulas.

We define the other standard logical operators in the usual way:

- f/y $=_{df}$ ¬(¬f & ¬y), f Æ y $=_{df}$ ¬(f & ¬y), f ´ y $=_{df}$ (f Æ y) & (y Æ f), and

- " af $=_{df}$ ¬$a¬f, []f $=_{df}$ ¬‡¬f.

Interpretations

Given a model, **M**, and a language, L, for ontology, we can now specify how L is interpreted in **M**. This is done in terms of an *interpretation function*, V, which maps elements of L to appropriate semantic objects of **M**. The general notion of an interpretation function is discussed at length in [Menzel 91a], so we will not dwell on details.[14] An interpretation function V for L and M = ·D,W,@,d,£,K,R,pÒ is a function such that:

- if t is a term (of L), then V(t) Œ D; $^{n}_{2}$,

- if P is an n-place first-order predicate (constant or variable) and wŒW, then V(P,w) Œ √(Dn); in particular, V(P^2_1,w) = {·a,aÒ | aŒd(w)}, and V(P^2_2,w) = £(w) (recall, P^2_1 is to be the identity predicate and P^2_2 the predicate that expresses the part-whole relation), and

- V(KIND) Œ √(K); V(DP) = {·k,pÒ | k Œ K and p Œ p(k)}; and V(SER_n) = {·r,k_1, ..., k_nÒ | ·k_1, ..., k_nÒ Œ p(r)}.

Interpretations for formulas of L will be defined recursively in terms of V in the usual way. Specifically, we define V´ to be a total extension of V such that V´ also maps the formulas of L into the set {T,F} (truth and falsity) in the following ways.

---

[14]C. Menzel and R. Mayer, "Theoretical Foundations for Information Representation and Constraint Specification," Armstrong Laboratory Report AL-TP-1991-0044, October 1991.

- If j is a first-order atomic formula Pt_1, ..., t_n, then V′(j,w) = T iff ·t_1, ..., t_nÒ Œ V(P,w).

- If j is KIND(P), then V′(j,w) = T iff V(P) Œ K.

- If j is DP(P,Q), then V′(j,w) = T iff V(P,w) Œ K and V(Q,w) Œ p(V(P,w)).

- If j is SER_n (P,P_1, ..., P_n), then V′(j,w) = T iff V(P) Œ R and ·V(P_1,w), ..., V(P_n,w)Ò Œ p(V(P,w)).

- If j is ¬y, then V′(j,w) = T iff V′(j,w) = F.

- If j is (y & q), then V′(j,w) = T iff both V′(y,w) and V′(q,w) = T.

- If j is $ay, then V′(j,w) = T iff there is a total extension V′′ of V differing from V′ at most in what it assigns to a such that V′′(y,w) = T.

- If j is ‡y, then V′(j,w) = T iff there is a w′ Œ W such that V′(y,w′) = T.

Axioms for Ontology

A proper axiomatic basis that captures the logic of our ontology models will be needed as a basis for developing computational tools with a capacity for automated reasoning. In this section, we will describe an appropriate axiomatic basis, though we will not explore the issues of computational implementation; that will be a task for the next phase of IDEF5 development.

The basis for the system will be a fairly weak second-order modal logic. That is, in addition to the usual propositional tautologies, and axioms for quantifiers and identity,[15] we also have the usual axioms of the modal logical system **S5**:

K: [](j .. y) .. ([]j .. []y),

[15]See again [Menzel 91a]. It should be noted that the logic developed in [Menzel 91a] is not second-order; however, the quantifier axioms for the logic in this report will work in exactly the same way, regardless of whether the quantified variable is first- or second-order.

T: j .. ‡j, and

5: ‡j .. []‡j;

and the rule of inference of *necessitation*:

Nec: If ¯æ j, then ¯æ []j.

That is, if a formula j is provable, then the proposition []j that it is necessary is also provable. This rule captures the intuition that anything provable is a truth of logic and hence should be true in all possible worlds or for all possible system states.[16]

The last thing we need are axioms that capture the logical content of the distinguished predicates of the ontology language--i.e., KIND, DP, and the predicates SER_n--that was given them in the definition of an interpretation above. Thus, we have:

O1: DP(F,G) .. (KIND(F) & ¬KIND(G)),

O2: DP(F,G) .. " x(Fx .. Gx),

O3: SER_n (F,F_1, ..., F_n) .. (KIND(F_1) & ... & KIND(F_n)),

O4: SER_n (F,F_1, ..., F_n) .. [(xF_1$x & ... & xF_n$x) .. (x_1F_1x_1$ & ... & xF_nx_n$ ‚ Fx$_1$...x$_n$)],

O5: DP(F,G) .. []DP(F,G), and

O6: SER_n (F,F_1, ..., F_n) .. []SER_n (F,F_1, ..., F_n).

O1 says that if the relation DP holds between two properties F and G, F must be a kind and G must not. This captures the idea that DP holds between a kind and any of its defining properties, which includes the idea that no kind is a defining property of any kind.

---

[16]There is some doubt about the soundness of necessitation as a general modal rule of inference; however, cf. C. Menzel, "The True Modal Logic," forthcoming in the *Journal of Philosophical Logic*.

O2 states that a member of a kind must have all the defining properties of the kind; it does *not* say that the member must have them *essentially* (in line with our earlier discussion of the notion of a kind). **O3** notes that system-essential relations are such relative to some finite collection of kinds. **O4** states that a system-essential relation must hold between representatives of the kinds it is relative to whenever these kinds are nonempty. **O5** and **O6** capture two important modal properties of kinds and system-essential relations: (1) if G is a defining property of a kind, F, it is *necessarily* a defining property of F; and (2) if F is system-essential relative to some collection of kinds, it is *necessarily* system-essential relative to those kinds. These properties are enforced in the model theory by the fact that the values of the function p on a member of K or R were defined independent of W. (Recall that p determines the defining properties of a given kind and the kinds relative to which a given relation is system-essential.) Note also that few properties of kinds and system-essential relations follow from the restriction that p be one-to-one. These are not expressible in the language as it stands because it requires that we be able to express identity between properties and relations; this, in turn, requires a second-order identity predicate. Addition of such a predicate will be explored in the next phase of IDEF5 development.

Bibliography

[Barwise 83] Barwise, J. and Perry, J., *Situations and Attitudes*, The MIT Press, Cambridge, 1983.

[Burch 91] Burch, R., *A Peircean Reduction Thesis: The Foundations of Topological Logic*, Texas Tech University Press, Lubbock, Texas, 1991.

[Devlin 91] Devlin, K., *Logic and Information, Volume I: Situation Theory*, Cambridge University Press, 1991.

[Hobbs 87] Hobbs, J., Croft, W., Davies, T., Edwards, D., and Laws, K., *The TACITUS Commonsense Knowledge Base*, Artificial Intelligence Research Center, SRI International, 1987.

[Link 83] Link, G., "The Logical Analysis of Plurals and Mass Terms: A Lattice Theoretic Approach," in R. Bauerle *et al.* (eds), *Meaning, Use, and Interpretation*, Berlin, De Gruyter, 1983.

[Menzel 90] Menzel, C., "Actualism, Ontological Commitment, and Possible World Semantics," *Synthese* **85** (1990), 355-389.

[Menzel 91a] Menzel, C., and Mayer R., "Theoretical Foundations for Information Representation and Constraint Specification," Air Force Armstrong Laboratory Report AL-TP-1991-0044, October 1991.

[Menzel 91b] Menzel, C., Mayer, R., and Edwards, D., "IDEF3 Process Descriptions and Their Semantics," forthcoming in Kuziak, A., and Dagli, C. (eds.), *Knowledge Based Systems in Design and Manufacturing*, Chapman Publishing, forthcoming 1991.

[Menzel 91c] Menzel, C., "The True Modal Logic," *Journal of Philosophical Logic* 20 (1991), 331-374.

[Sowa 84] Sowa, J., *Conceptual Structures: Information Processing in Mind and Machine*, Addison-Wesely Publishing Co., Reading, MA, 1984.

[Webster 88] *The Merriam-Webster Dictionary*, Simon & Schuster, New York, NY, 1986.

☆U.S. Government Printing Office: 1993 – 750-061/60241