



May 1982

The ENHANCE System: Creating Meaningful Sub-Types in a Database Knowledge Representation for Natural Language Generation

Kathleen Filliben McCoy
University of Pennsylvania

Follow this and additional works at: https://repository.upenn.edu/cis_reports

Recommended Citation

Kathleen Filliben McCoy, "The ENHANCE System: Creating Meaningful Sub-Types in a Database Knowledge Representation for Natural Language Generation", . May 1982.

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-82-06.

This paper is posted at ScholarlyCommons. https://repository.upenn.edu/cis_reports/647
For more information, please contact repository@pobox.upenn.edu.

The ENHANCE System: Creating Meaningful Sub-Types in a Database Knowledge Representation for Natural Language Generation

Abstract

The knowledge representation is an important factor in natural language generation since it limits the semantic capabilities of the generation system. It is, however, a tedious task to hand code a knowledge representation which reflects both a user's view of a domain and the way that domain is modelled in the database. A system is presented which uses the contents of the database to form part of a database knowledge representation automatically. It augments a database schema depicting the database structure used for natural language generation. Computational solutions are presented for deriving the information types contained in the schema. Three types of world knowledge axioms are used to ensure that the representation formed is meaningful and contains salient information.

Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-82-06.

UNIVERSITY OF PENNSYLVANIA
THE MOORE SCHOOL OF ELECTRICAL ENGINEERING
SCHOOL OF ENGINEERING AND APPLIED SCIENCE

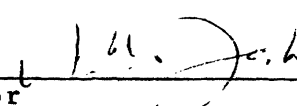
THE ENHANCE SYSTEM:
CREATING MEANINGFUL SUB-TYPES IN A DATABASE KNOWLEDGE
REPRESENTATION FOR NATURAL LANGUAGE GENERATION

Kathleen Filliben McCoy

Philadelphia, Pennsylvania

May 1982

A thesis presented to the Faculty of Engineering and Applied
Science of the University of Pennsylvania in partial
fulfillment of the requirements for the degree of Master of
Science in Engineering for graduate work in Computer and
Information Science.



Supervisor



Graduate Group Chairman

ABSTRACT

The ENHANCE system:
Creating Meaningful Sub-Types in a Database Knowledge
Representation For Natural Language Generation

Kathleen Filliben McCoy

SUPERVISOR: Aravind K. Joshi

The knowledge representation is an important factor in natural language generation since it limits the semantic capabilities of the generation system. It is, however, a tedious task to hand code a knowledge representation which reflects both a user's view of a domain and the way that domain is modelled in the database. A system is presented which uses the contents of the database to form part of a database knowledge representation automatically. It augments a database schema depicting the database structure used for natural language generation. Computational solutions are presented for deriving the information types contained in the schema. Three types of world knowledge axioms are used to ensure that the representation formed is meaningful and contains salient information.

ACKNOWLEDGEMENTS

I would like to thank my advisor Aravind K. Joshi for his guidance throughout the course of this work.

Special thanks also goes to Kathleen R. McKeown for her many ideas, patience, and direction throughout this research. It could not have been done without her help.

Bonnie Webber deserves special thanks for her many comments concerning this work.

I would also like to thank many students in the Moore School for their support during this work, especially Eric Mays, Sitaram Lanka, Steve Bossie, and the entire north corridor.

Finally I would like to thank my family for their patience and unending support. Special thanks goes to my husband, Scott, who has been extremely helpful and understanding. He helped make this all possible.

This work was partially supported by National Science Foundation grant #MCS81-07290.

Table of Contents

1.0	INTRODUCTION	1
2.0	GENERATING SUB-TYPES	6
3.0	OVERVIEW OF THE KNOWLEDGE REPRESENTATION	9
3.1	Representation Below Database Entities	15
4.0	WORLD KNOWLEDGE AXIOMS	19
4.1	Very Specific Axioms	22
4.2	Specific Axioms	27
4.2.1	Forming Breakdowns	31
4.2.2	Selecting Salient DDAs	34
4.2.3	Specific Axiom Conclusions	37
4.3	General Axioms	38
4.3.1	Naming Conventions	40
4.3.1.1	Summary	43
4.3.2	CLASS-OTHER Formation	44
4.3.2.1	Summary	49
4.3.3	Rules For Accepting A Breakdown	50
4.4	Conclusions	54
5.0	DESCRIPTIVE INFORMATION (IMPLEMENTATION PRINCIPLES)	57
5.1	System Overview	57
5.2	Based DB Attribute	59
5.3	Distinguishing Descriptive Attributes (DDAs)	61
5.4	Constant DB Attributes	68
5.5	Constant Relation Attributes	70
5.6	Subset Entities List	70
6.0	USE OF KNOWLEDGE REPRESENTATION GENERATED BY ENHANCE	74
7.0	FUTURE WORK	79
8.0	CONCLUSION	81
9.0	REFERENCES	83

APPENDIX A LIST OF ATTRIBUTES ASSOCIATED WITH EACH ENTITY

APPENDIX B BREAKDOWNS CREATED BY ENHANCE

APPENDIX C SAMPLE NODE INFORMATION CREATED BY ENHANCE

1.0 INTRODUCTION

As the use of database systems by non-trained personnel becomes widespread, it is increasingly important that the knowledge needed to extract meaningful information from the database system is easily obtained. An optimal way of acquiring this knowledge is to converse, in natural language, with the system itself. It has been found ([Malhotra 75], [Tennant 79]) that one important kind of question that people often ask in order to familiarize themselves with the database, are questions about the database structure. The TEXT system [McKeown 82] was developed to answer these types of questions.

Before a system can take advantage of TEXT, its knowledge about itself must be rich enough to support the generation of natural language text. Since time is an important factor in the generation process, the knowledge representation must contain all (or most) of the information needed for an answer in order to avoid extensive inferencing. The ENHANCE system has been developed to augment the database schema used by TEXT so that richer descriptions of the database can be generated.

Introduction

A desirable feature in any generation system is that it be portable. One major bottleneck in the portability of such systems is the knowledge representation. Moving the generation system from one domain to another usually requires hand coding the entire knowledge representation over again. The ENHANCE system alleviates much of this problem by automatically creating part of the knowledge representation based on the contents of the database. This relieves the user of the tedious job of generating the entire representation by hand. The only input required to the ENHANCE system is a set of world knowledge axioms which are formulated in such a way as to employ database concepts. Thus, the input can be easily provided by the database manager.

The TEXT system, used to give text length responses to questions about database structure, handles three types of questions:

1. requests for the definition of an entity (What is an <el>?)
2. requests for the information available about an entity (What do you know about <el>?)

3. requests concerning the difference between two entities (What is the difference between <e1> and <e2>?)

In order to answer these questions, the knowledge representation used by TEXT contains several features used in standard database models. It consists of a meta-level description of the database based on the Chen entity-relationship model [Chen 76] and the generalization principles used by the Smith's [Smith & Smith 77] and Lee and Gerritsen [Lee & Gerritsen 78]. There is a generalization hierarchy on the entities; each node in the hierarchy contains descriptive information needed for the generation process.

The ENHANCE system augments the knowledge representation by creating information about sub-types of the entities for which physical records exist in the database (database entity classes). ENHANCE infers sub-types and generates all descriptive information associated with the sub-types using the actual database values. The world knowledge axioms ensure that the generated sub-types are meaningful and that salient information is chosen for their descriptions. The ENHANCE system is run only once for a particular database. The resulting representation can be used by the generation

Introduction

system on all subsequent queries. The goal of the ENHANCE system is to generate a meta-level description of the database structure which reflects both the user's view of the domain and the way that domain is modelled by the database. Using a system for this purpose relieves the generation system of extensive inferencing and relieves the database manager of the tedious job of creating the entire knowledge representation by hand.

Creating this sub-type information before it is actually needed by the generation system does have some space/time tradeoffs. After ENHANCE is run, the knowledge representation is considerably longer. However, the generation system is now able to handle questions requiring information about sub-types in a minimal amount of time. Since the generation system must be concerned with the amount of time it takes to answer a question, the cost in space used for the large knowledge representation is well worth its savings in inferencing time. If, however, at some future point, time is no longer a major factor in natural language generation, many of the ideas put forth here could be used to generate sub-type information only as it is needed.

Introduction

The approach taken to sub-type generation will first be discussed. This is followed by a description of the TEXT database model. Next, the world knowledge axioms will be presented as the solution to some problems encountered by a system which augments a knowledge representation. Next some principles used in implementing ENHANCE will be presented followed by some sample uses of the representation formed and some future directions.

Generating Sub-types

2.0 GENERATING SUB-TYPES

Recall that TEXT uses a generalization hierarchy on the entities. It was assumed that this hierarchy would be hand coded by the database designer. In this work, the level in the hierarchy corresponding to the database entity classes is identified. Since the hierarchy above this level is based almost entirely on world knowledge, it is assumed that it must be hand coded. There is information contained in the database itself, however, which can be used to create the hierarchy below the level of the database entity classes automatically.

The approach to sub-type creation taken by ENHANCE is that laid out by Smith and Smith [Smith & Smith 77] and followed by Lee and Gerritsen [Lee & Gerritsen 78]. That is, using the observation that each attribute that an entity class possesses can serve to partition that entity class into a number of mutually exclusive sub-types (sub-classes). For example, in a database containing PEOPLE, attribute SEX can be used to partition the instances of PEOPLE into two mutually exclusive sets: MALE and FEMALE.

Generating Sub-types

Some partitions of the entity class are more informative than others. Above, if all of the instances of PEOPLE in the database had SEX = FEMALE, that partition would not be very informative. The information it provides can already be derived from the representation, since the (one) sub-class would simply reflect the entity class as a whole. A partition based on the attribute used as the primary key would also not yield a very interesting partition. In this case, there would be one sub-class for each instance in the database. Thus, the sub-classes would add no information which is not derivable from the database itself.

The ENHANCE system uses a set of world knowledge axioms to ensure that the attributes used to partition the entity classes yield meaningful sub-types. They help in two ways: 1) they guide the system in choosing the attributes to use as the basis for a breakdown, 2) they ensure that the resulting partitions are informative. The world knowledge axioms are discussed in detail in chapter 4 after first describing the database model used by TEXT.

As mentioned above, for each sub-type resulting from a partition, a node is created in the generalization hierarchy. This node must contain information needed for the generation process indicating how a sub-type differs from its siblings. This information is created by ENHANCE

Generating Sub-types

by comparing the values of attributes within the sub-types. ENHANCE uses the world knowledge axioms to record the major and most salient differences between the sub-types. This information is used by the generation system to make comparisons (analogies) among the sub-types.

Overview of the Knowledge Representation

3.0 OVERVIEW OF THE KNOWLEDGE REPRESENTATION

The knowledge representation used by the TEXT system [McKeown 82] is a meta-level description of the database based on the Chen entity-relationship model [Chen 76] and the generalization principles of Smith and Smith [Smith & Smith 77]. In addition to the items found in these standard database models, it includes several pieces of descriptive information to provide a "real world" view of the database.

The knowledge representation consists of a generalization hierarchy based on the database entity classes. Each node in the hierarchy has a unique name, attributes, relations, descriptive information used for the generation process, and links to both its immediate parents and descendents. (There is also a hierarchy on the database attributes termed the topic hierarchy.) Each node in the generalization hierarchy is either a generalization or a specialization a database entity class.

def 3.1 - database entity class - class of database instances for which physical records exists. These instances have common database attributes and relations associated with them.

def 3.2 - database entity generalization - generalization of an actual database entity class - usually depicts the common features of a number

Overview of the Knowledge Representation

of database entity classes.

def 3.3 - database entity subset - specialization of a database entity class - some subset of the instances which make up the entity class.

def 3.4 - entity - common name referring to either a database entity class, database entity generalization, or a database entity subset.

For example, the database entity classes SHIP and SUBMARINE are generalized as the entity WATER-VEHICLE. Entities WATER-VEHICLE and AIR-VEHICLE are generalized as entity VEHICLE. Thus WATER-VEHICLE is termed the superordinate of both SHIP and SUBMARINE. SHIP and SUBMARINE are termed mutually exclusive sub-types of WATER-VEHICLE and are siblings of each other. Figure 3.1 shows part of the hierarchy used by the TEXT system for the ONR database. The portion shown depicts the database entity classes and their generalizations.

Overview of the Knowledge Representation

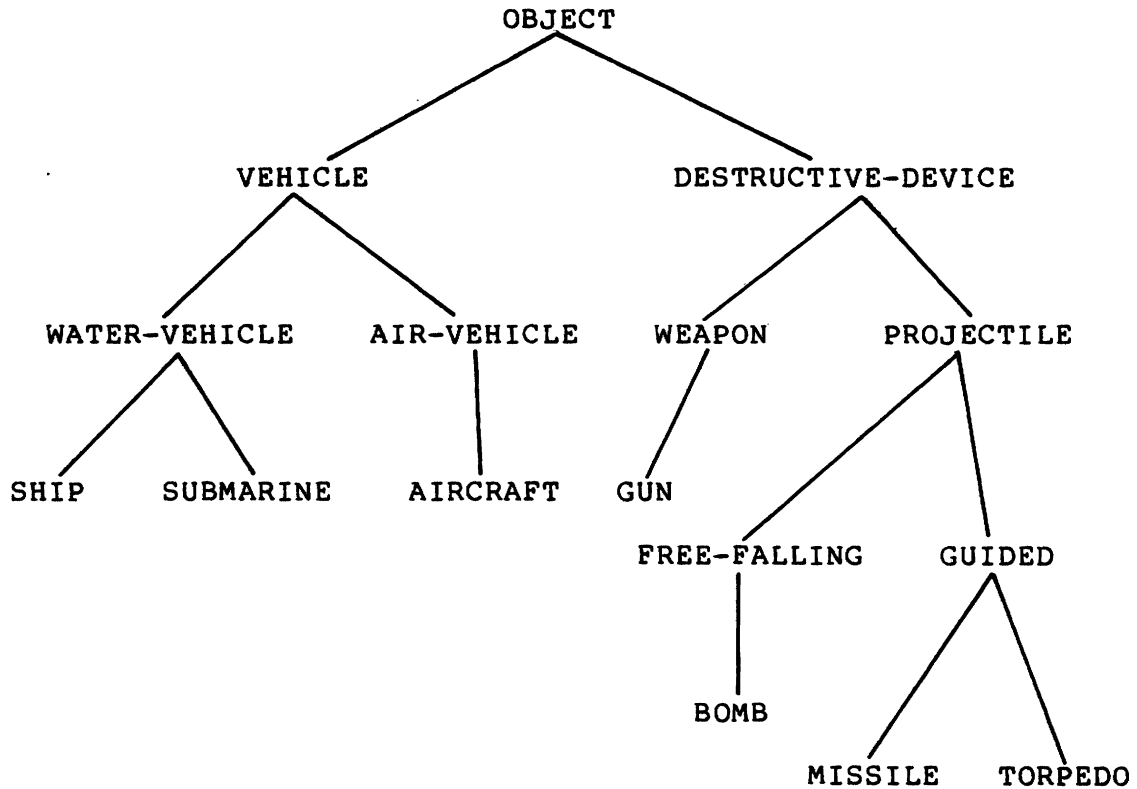


Figure 3.1 Generalization Hierarchy Above Database Entities

TEXT uses the generalization hierarchy to define or to provide information about entities in terms of 1) their constituents (e.g. "There are two types of entities in the ONR database: destructive devices and vehicles."*); 2) their superordinates (e.g. "A destroyer is a surface ship ... A bomb is a free falling projectile." and "A whiskey is an underwater submarine.").

* the quoted material is excerpted from actual TEXT output.

Overview of the Knowledge Representation

The database attributes are attached to the hierarchy at the highest level possible; all descendents of an entity inherit the attributes which are attached to the entity. Associated with each attribute is a constraint on its values. For example, a constraint may specify that a SHIP has attribute LENGTH which must be a number greater than 0. The attribute information is used by TEXT to identify information associated with an entity and to compare entities by contrasting their attribute information. For example, "Other DB attributes of the missile include PROBABILITY_OF_KILL, SPEED, ALTITUDE ... Other DB attributes of the torpedo include FUSE_TYPE, MAXIMUM_DEPTH, ACCURACY_&_UNITS...".

The knowledge representation contains both generic relations and instances of relations. A relation instance is a relation occurring in the database between two particular entities. A generic relation is a generalization of a set of relation instances. For example, the ON relation in the ONR database holds between SHIPS and MISSILES, AIRCRAFT and GUNS, etc... The generic relation, ON, in the knowledge representation captures the information about the relation common to each instance of the relation. This information includes the functionality of the relation and any attributes that are associated with the relation. An instance of a relation, on the other hand, just captures

Overview of the Knowledge Representation

the information about the particular occurrence of the generic relation. Associated with a relation instance is the unique instance name, the corresponding generic name, and the names of the two entities participating in the instance with their allocated roles. The relational information is used by TEXT to compare entities participating in different instances of a common generic relation. For example, since both missiles and torpedoes participate in the same generic relation, the following comparison is made by TEXT: "Missiles are carried by water-going vehicles and aircraft .. Torpedoes are carried by water-going vehicles."

In addition to the above information which is found in other database models, the knowledge representation contains two types of information which provide additional descriptive power. The first of these is termed a distinguishing descriptive attribute (DDA). This is an attribute (not necessarily an actual database attribute) which is associated with a split in the hierarchy. It indicates the real world reason for the split. Each mutually exclusive sub-type resulting from a split in the hierarchy will have the same DDA name, the value of the DDA will distinguish one sub-type from another. For example, an OBJECT is broken down into two mutually exclusive sets: VEHICLES and DESTRUCTIVE-DEVICES. Associated with this

Overview of the Knowledge Representation

split is the DDA FUNCTION. The VEHICLE has FUNCTION = TRANSPORTATION while the DESTRUCTIVE-DEVICE has FUNCTION = LETHALITY. TEXT uses this information to identify major descriptive characteristics of an entity. Examples include: "A guided projectile is a projectile that is self-propelled." and "A ship is a water-going vehicle that travels on the surface."

There is also a set of actual database attributes associated with each split in the hierarchy. These are termed supporting DB attributes since they support the choice of the DDA used for each entity. These are attributes which actually occur in the database that provide actual DB evidence indicating the basis for the split. These attributes are similar to what Lee and Gerritsen term partition-attributes (p-attributes) [Lee & Gerritsen 78]. The p-attribute is an actual database attribute whose value is used to partition the entity into a number of mutually exclusive sub-classes. It was found that in this application, it was not always possible to find a single database attribute whose value could be used to partition the entity. At the higher levels of the hierarchy, the entities are sub-divided according to the different attributes they possess. These attributes re-enforce the DDA chosen for the split. In the example given above, the VEHICLE's DDA is supported by the fact that all VEHICLES in

Overview of the Knowledge Representation

the database have some type of travel-means and speed-indices. The DESTRUCTIVE-DEVICE's DDA, on the other hand, is supported by the occurrence of some type of lethal-indices in the DB attributes list of all DESTRUCTIVE-DEVICES. Examples of the kind of information provided by the supporting DB attributes include: "Its (the ship's) surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT." and "The guided projectile's propulsion capabilities are provided by the DB attributes under SPEED_INDICES (for example, MAXIMUM_SPEED) and FUSE_TYPE."

3.1 Representation Below Database Entities

The information available below the level of the database entity classes is somewhat different from that available above this level in the hierarchy. Since all of the database attributes are present at the level of the database entities, the values that the attributes take on becomes important below this level.

Below the level of the database entities an actual database attribute can be found which uniquely identifies an instance of a database entity as belonging to a particular sub-class. This attribute and its associated value are termed the based DB attribute. This is the counterpart of the supporting DB attribute above the database entity level.

Overview of the Knowledge Representation

It is related to the partition attribute of Lee and Gerritsen in that its values define a set of sub-classes. For example, the sub-class KITTY-HAWK-SHIP is defined as the set of instances of database entity SHIP whose value for attribute CLASS = KITTY-HAWK. Thus, the based DB attribute for KITTY-HAWK-SHIP is (CLASS = KITTY-HAWK). The based DB attribute may be in the form of a disjunction or may specify only a part of an attribute value field. This can be seen from the based DB attribute for SHIP sub-type CRUISER. The CRUISER is defined to be a SHIP whose first two characters of attribute HULL-NO are CA or CG or CL. TEXT uses this information to indicate why an individual falls into one sub-type as opposed to another. For example, "A submarine is classified as a whisky if its CLASS is WHISKY." and "A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_NO are CV."

Below the level of the database entities it is also important to associate a DDA with each sub-type. This must exhibit a descriptive distinction between sub-types (rather than a defining difference as exhibited in the based DB attribute). Below the database entity class level, this distinction takes the form of a set of actual DB attributes whose collective value differentiates a particular sub-class from all other sub-classes in the breakdown. For example, since an AIRCRAFT-CARRIER has a LENGTH greater than any

Overview of the Knowledge Representation

other type of SHIP, the DDA of AIRCRAFT-CARRIER can be LENGTH. The value of the DDA for AIRCRAFT-CARRIER is the range of values that the attribute takes on within the sub-class (in this case it would be 1039 - 1063). It should be noted that it may in general take more than one attribute to distinguish on sub-type from the rest. This is the case for sub-type AMPHIBIOUS-AND-LANDING-SHIP whose DDA is the set of attributes (MAXIMUM-SPEED and LENGTH). Two attributes are required since some other types of ships have the same MAXIMUM-SPEED as the AMPHIBIOUS-AND-LANDING-SHIP, while others have the same LENGTH. TEXT uses the DDA to exhibit the most salient distinctions of the sub-types. For example, "An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063." and "Echo IIs have a PROPULSION_TYPE of NUCL and a FLAG of RDRD."

Other information is added below the entity level to allow richer sub-type comparisons by the generation system. For example, if a database attribute or relation attribute has a constant value throughout a sub-type, this value is recorded. Ranges of values of attributes may also be recorded in one sub-type if the attributes are used as DDAs for a mutually exclusive sibling (a sibling resulting from the same breakdown). This allows the generation system to show how the attributes included in the DDA of one sub-type

Overview of the Knowledge Representation

differ from the same attributes of another sub-type. For example, TEXT is able to make the following simple inference: "Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships.". This inference is easily made since the values of the attributes appearing in the DDA of aircraft carriers are recorded in the DB attributes list of each of its sibling sub-classes. The values of relational attributes are also useful in making comparisons between sub-types. For example, "Ocean escorts carry between 2 and 22 torpedoes, 16 missiles and between 1 and 2 guns ... Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles and between 1 and 4 guns.".

See chapter 6 for further examples of TEXT using the representation below the database entity classes created by ENHANCE.

4.0 WORLD KNOWLEDGE AXIOMS

In order for the generation system to generate meaningful descriptions of the database, the knowledge representation must effectively capture both the user's view of the database and the actual values in the database. The danger of automatically generating pieces of the knowledge representation is that the resulting representation may fail to capture the user's view of the database. There must be some notion of real world knowledge in order to make sure that the breakdowns generated are meaningful. With no account of this real world knowledge, there are several ways in which an automatically generated representation may deviate from a user's expectations. One way is that the representation may fail to capture the user's preconceived notions of how a certain database entity should be broken down into sub-classes. This would occur if these preconceived breakdowns were not solely based on an attribute present in the database. For instance, the breakdown may be based on just parts of an attribute value field. If this were the case, there would be no way for the system to generate such a breakdown without information mapping the important parts of the particular attribute value field into the desired sub-type names. There should

World Knowledge Axioms

be some way of including this type of information, since the resulting breakdowns would be very meaningful to the user.

A representation may also deviate from a user's expectations if inappropriate attributes are used to partition an entity class. Clearly, some attributes are more salient than others. It would seem very natural to have a breakdown of SHIP based on attribute CLASS, but one based on attribute FUEL_CAPACITY would seem less likely. A partition based on CLASS would yield sub-classes like SKORY and KITTY-HAWK, while one based on FUEL_CAPACITY could only yield ones like SHIPS-WITH-100-FUEL-CAPACITY. Since saliency is not an intrinsic property of an attribute, there must be some way of indicating attributes salient in a domain. Breakdowns based on these attributes would be more informative to the user since they would reflect preconceived breakdowns of a user familiar with the domain.

Once breakdowns have been made, the descriptive information for the sub-classes must be chosen. Here the importance of choosing salient attributes is crucial. Even though a DESTROYER may be differentiated from other types of ships by its ECONOMIC-SPEED, it seems more informative to distinguish it in terms of the more commonly mentioned property DISPLACEMENT. The descriptive information of a sub-type should be chosen from salient information if possible.

World Knowledge Axioms

A final problem faced by a system which only relies on the database contents is that a partition formed may be essentially meaningless (adding no new information to the representation). This can occur if all of the instances in the database fall into either the same sub-class or if each one falls into different sub-classes. Such breakdowns either exactly reflect the entity class as a whole, or reflect the individual instances. This same type of problem occurs if the only difference between two sub-classes is the attribute the breakdown is based on. That is, when the only real difference between two different sub-classes is their based DB attribute. Thus, the attribute chosen for the breakdown exerts no influence over the other attributes. Such a breakdown would add no information that could not be trivially derived from the database itself.

ENHANCE handles the above problems by using a set of world knowledge axioms. The axioms guide ENHANCE to ensure that the breakdowns formed are appropriate and that salient information is chosen for the sub-class descriptions. At the same time, the axioms give the user control over the representation formed. The axioms can be changed and the system rerun. The new representation will reflect the new set of world knowledge axioms. In this way the user can tune the representation to his/her needs.

World Knowledge Axioms

The ENHANCE system uses three types of world knowledge axioms: very specific to the database, specific to the domain, and general. The categories reflect the extent to which the axioms must be changed when moving the system from one database to another. Each axiom category, how they are used by ENHANCE, and the problems each category solves will be discussed below.

4.1 Very Specific Axioms

The very specific axioms give the user the most control over the representation formed. In fact, they let the user specify breakdowns that s/he would a priori like to appear in the knowledge representation. The axioms are formulated in such a way as to allow breakdowns on parts of the value field of a character attribute, and on ranges of values for a numeric attribute (examples of each are given below). This type of breakdown could not be formed without explicit information mapping the defining portions of the attribute value field into the desired sub-type names. This semantic mapping can not be derived from the database alone.

A sample use of the very specific axioms can be found in classifying ships by their type (i.e. aircraft-carriers, destroyers, mine-warfare-ships, etc...). In military dictionaries (see [Blackman 73] and [Carrison 68]) this is a very common breakdown of ships. Assuming there is no

database attribute which explicitly gives the ship type, with no additional information there is no way of generating that breakdown of ship. The partition can be derived, however, if a semantic mapping between the sub-type names and existing attribute value pairs can be identified.

A user knowledgeable of the domain would note that there is a way to derive the type of a ship based on its HULL_NO. In fact, the first one or two characters of the HULL_NO uniquely identifies the ship type. For example, all aircraft-carriers have a HULL_NO whose first two characters are CV, while the first two characters of the HULL_NO of a CRUISER are CA or CG or CL. This linking of the ship type with the defining portions of the HULL_NO can be accomplished using a very specific axiom. An example of such an axiom is shown in Figure 4.1. This was an actual specific axiom used by the ENHANCE system to generate the breakdown of the entity SHIP into its various ship types.

World Knowledge Axioms

```
(SHIP "SHIP_HULL_NO"  
      "OTHER-SHIP-TYPE"  
      (1 2 "CV" "AIRCRAFT-CARRIER")  
      (1 2 "CA" "CRUISER")  
      (1 2 "CG" "CRUISER")  
      (1 2 "CL" "CRUISER")  
      (1 2 "DD" "DESTROYER")  
      (1 2 "DL" "FRIGATE")  
      (1 2 "DE" "OCEAN-ESCORT")  
      (1 2 "PC" "PATROL-SHIP-AND-CRAFT")  
      (1 2 "PG" "PATROL-SHIP-AND-CRAFT")  
      (1 2 "PT" "PATROL-SHIP-AND-CRAFT")  
      (1 1 "L" "AMPHIBIOUS-AND-LANDING-SHIP")  
      (1 2 "MC" "MINE-WARFARE-SHIP")  
      (1 2 "MS" "MINE-WARFARE-SHIP")  
      (1 1 "A" "AUXILIARY-SHIP"))
```

Figure 4.1 Very Specific Axiom for Character Attribute

The axiom in Figure 4.1 is an example of a very specific axiom which maps parts of a character attribute value field into the sub-type names. The axiom gives the system several pieces of information needed to create the breakdown. The first field of any very specific attribute specifies the database entity class that the axiom addresses. The axiom above addresses the entity SHIP. The second field specifies the attribute the axiom uses (HULL_NO in this case). The third field specifies the "class-other-name". This is the name of the sub-class containing any ships which do not fit into one of the specified categories. (Class-other is discussed in detail in section 4.3.2.) The remaining fields indicate the mapping

from specific values in the attribute field to the sub-type names. For example, the first such field is read: If characters one through two of the HULL_NO = CV then put the instance in sub-type AIRCRAFT-CARRIER. The field gives the starting character position, the ending character position, the value of the partial field, and the resulting sub-type name. In the ONR database instances of each type of ship are present. Therefore, application of the above axiom results in a breakdown of SHIP containing the nine sub-classes (or sub-types) specified.

Sub-typing of entities can also be specified on the basis of the ranges of values of a numeric attribute. For example, the entity BOMB is often sub-typed by the range of the attribute BOMB_WEIGHT. A bomb is classified as being HEAVY, MEDIUM-WEIGHT, or LIGHT-WEIGHT. An axiom which specifies this (for the bombs found in the ONR database) is shown in FIGURE 4.2.

```
(BOMB "BOMB WEIGHT"  
  "OTHER-WEIGHT-BOMB"  
  (900 99999 "HEAVY-BOMB")  
  (100 899 "MEDIUM-WEIGHT-BOMB")  
  (0 99 "LIGHT-WEIGHT-BOMB"))
```

Figure 4.2 Very Specific Axiom for Numeric Attribute

World Knowledge Axioms

Since this axiom refers to an attribute with a numeric value, the range of the attribute value is delineated for each sub-type. In this case, the first field which specifies the sub-type is read: If attribute BOMB_WEIGHT is between 900 and 99999 then the bomb is classified "HEAVY-BOMB".* The breakdown of BOMB generated by ENHANCE resulting from the very specific axiom shown above is depicted in Figure 4.3.

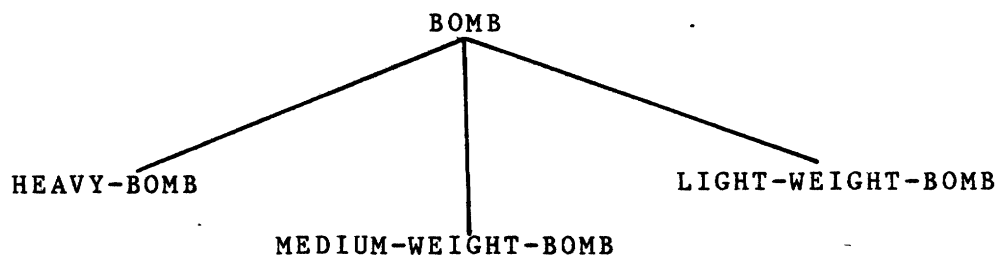


Figure 4.3 Breakdown of BOMB Based on Very Specific Axiom

Formation of the very specific axioms require in-depth knowledge of both the domain the database reflects, and the database itself. Knowledge of the domain is required in order to know common classifications (breakdowns) of objects in the domain. Knowledge of the database is needed in order to convey these breakdowns in terms of the database

* It is assumed here that all bomb-weights are expressed in the same units. This conversion is done by ENHANCE (see section 5.1).

attributes. It should be noted that this type of axiom is not required for the system to run. If the ENHANCE user has no preconceived ideas about what breakdowns should appear in the representation, no very specific axioms need to be specified.

The purpose of the very specific axioms is to give the ENHANCE user control over the representation formed. They enable him/her to specify breakdowns that s/he would a priori like to appear in the representation. These breakdowns may not be derivable from the database attributes alone; additional semantics may be needed to associate various sub-type names with attribute fields. The very specific axioms provide the user with the means for specifying breakdowns which would otherwise not appear in the automatically generated part of the representation.

4.2 Specific Axioms

The specific axioms afford the user less control than the very specific axioms, but are still a powerful device. The specific axioms are used to point out which database attributes are more salient (or more important to the domain) than others. They are used in various ways by the system. These range from pointing out which attributes to form breakdowns on, to suggesting which attributes to use as descriptive information for a sub-class.

World Knowledge Axioms

One of the most striking features of the specific axioms is their simplicity. In fact, the axioms consist of a single list of database attributes which are singled out as being important to the domain. The list is termed the important attributes list and is used to point out attributes which are usually referred to when discussing the domain the database reflects. The important attributes list does not "control" the system as the very specific axioms do. Instead it suggests paths for the system to try; it has no binding effects.

The important attributes list used for testing ENHANCE on the ONR database is shown in Figure 4.4. Notice that both character attributes and numeric attributes are included, and that at least one attribute is present for each entity in the database. The database entities include: SHIP, SUBMARINE, AIRCRAFT, BOMB, TORPEDO, and MISSILE. (See Appendix A for list of attributes associated with each entity.)

```
(CLASS FLAG
  DISPLACEMENT
  LENGTH
  WEIGHT
  LETHAL_RADIUS
  MINIMUM_ALTITUDE
  ACCURACY
  HORZ_RANGE
  MAXIMUM_ALTITUDE
  FUSE_TYPE
  PROPULSION_TYPE
  PROPULSION
  MAXIMUM_OPERATING_DEPTH
  PRIMARY_ROLE))
```

Figure 4.4 Important Attributes List

The list was constructed by examining texts (see [Blackman 73], [Carrison 68] and [Palmer 75]) about the domain the ONR database reflects, and noticing which attributes were referred to (directly or indirectly). These attributes were placed in the important attributes list. For example, it is very common in the literature to affiliate all entities in the Navy domain with their country. Thus, we refer to US ships, submarines, missiles, and aircraft and to Soviet ships, submarines, missiles, and aircraft. In the ONR database, the country of an entity is indicated by attribute FLAG. Since this affiliation is important when discussing the domain, FLAG appears in the important attributes list.

World Knowledge Axioms

Other attributes on the list may not be as "universal" (i.e. apply to as many entities) as attributes like FLAG. However, these attributes may still be important to discussing a particular entity. Examples of this type of attribute are also found on the important attributes list. Only the entity SHIP has attribute DISPLACEMENT. But, the DISPLACEMENT of a SHIP is often referred to when giving a definition of a specific type of SHIP; thus it is included on the list. The important attributes list, therefore, may include attributes which refer to either a single entity, or to many entities.

ENHANCE has two major uses for the important attributes list. First, ENHANCE attempts to form breakdowns based on some of the attributes in the list. Second, ENHANCE uses the list to decide which attributes are better distinguishing descriptive attributes (DDAs) (see section 5.3 for an explanation of DDAs) than others. Thus, ENHANCE uses the same list for guidance in two very different tasks. It must decide which attributes are better for basing breakdowns on and which are better for describing the resulting sub-classes. Most attributes important to the domain are good for descriptive purposes, but some attributes are better than others as the basis for a breakdown. Even though DISPLACEMENT is a very important attribute when discussing ships, one would not expect to see

a breakdown of SHIP with the following sub-classes:
200-DISPLACEMENT-SHIP, 1000-DISPLACEMENT-SHIP,
78000-DISPLACEMENT-SHIP, etc.....

4.2.1 Forming Breakdowns -

Some attributes that are better as the basis for a breakdown include CLASS, FLAG, and FUSE-TYPE. Attribute CLASS breaks SHIP into meaningful sub-classes (see Figure 4.5) while attribute DISPLACEMENT seemed awkward as the basis for a breakdown.

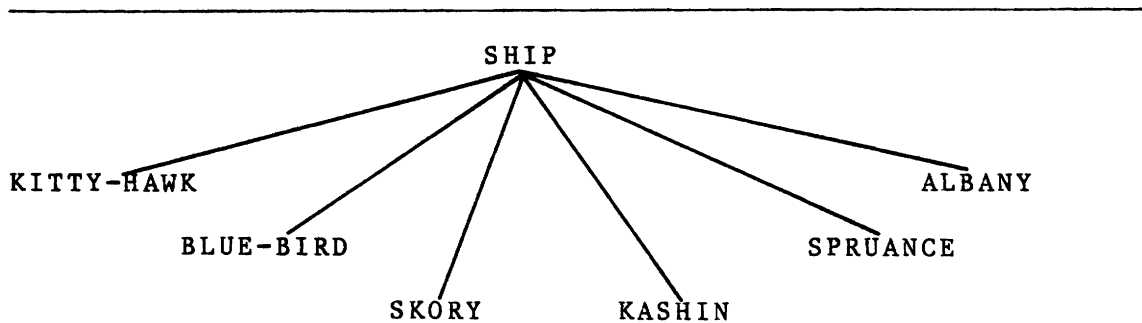


Figure 4.5 SHIP Sub-classes Based on Attribute CLASS

Attribute FUEL-TYPE breaks the SUBMARINE into common sub-classes (i.e. NUCLEAR-SUB, DIESEL-SUB, etc...), while MAXIMUM-OPERATING-DEPTH, although often discussed when talking about SUBMARINES, is rarely used as the basis of a breakdown. Some attributes, while important to the domain,

World Knowledge Axioms

are not suitable as the basis for a breakdown.

The common feature of such attributes is that they are numeric attributes. Attributes with character values can more naturally act as the basis for a breakdown. One reason for this is the finite nature of the values of a character attribute (as opposed to a numeric attribute). Since the number of integers is infinite, breaking up an entity on the basis of a numeric attribute could, in principle, lead to an infinite number of sub-classes. On the other hand, character attributes often have a small set of legal values. A breakdown based on such an attribute would lead to a small well defined set of sub-classes. This same distinction is made in the TEAM system [Grosz et. al. 82]. In discussing symbolic (character) attributes, TEAM is willing to talk about sub-classes of an entity class based on the value of that attribute (e.g. MCDONNELL AIRCRAFT - where MCDONNELL is a particular value for attribute MANUFACTURER). This is not permitted for numeric or boolean attributes.

ENHANCE uses this distinction between character attributes and numeric attributes when deciding which attributes to use as the basis for breakdowns. It first attempts to form breakdowns of an entity based on character attributes from the important attributes list. Only if all of these breakdowns fail (see section 4.3.3 for reasons for breakdowns failing), does the system attempt breakdowns

based on numeric attributes. Thus, two principles are used for attempting breakdowns: 1) character attributes are better as the basis for a breakdown; 2) it is better to have breakdowns based on numeric attributes than no breakdowns at all.

These ideas are illustrated in the breakdown formed for entity TORPEDO by the ENHANCE system. There are only two attributes in the important attributes list which pertain to entity TORPEDO. These are ACCURACY (a numeric attribute) and FUSE_TYPE (a character attribute). Using principle 1) above, ENHANCE attempts to form a breakdown based on attribute FUSE_TYPE. This will presumably lead to sub-classes like: IMPACT-FUSE-TORPEDO and TIMED-FUSE-TORPEDO. If this breakdown is accepted, no other breakdowns will be attempted. It just so happens that every torpedo in the ONR database has the same FUSE_TYPE (in particular IMPACT). Thus, only one sub-class is formed for entity class TORPEDO. When this is the case, the breakdown is not used by the system since it adds no new knowledge to the representation. Since the breakdown based on FUSE_TYPE is thrown out, there are no breakdowns of TORPEDO based on the attributes in the important attributes list. Using principle 2) above, ENHANCE goes back to the important attributes list and looks for numeric attributes of TORPEDO. Since ACCURACY is found, a breakdown based on ACCURACY is

World Knowledge Axioms

attempted. This breakdown succeeds and is therefore added to the knowledge representation. Figure 4.6 shows the resulting breakdown of entity TORPEDO generated by ENHANCE.

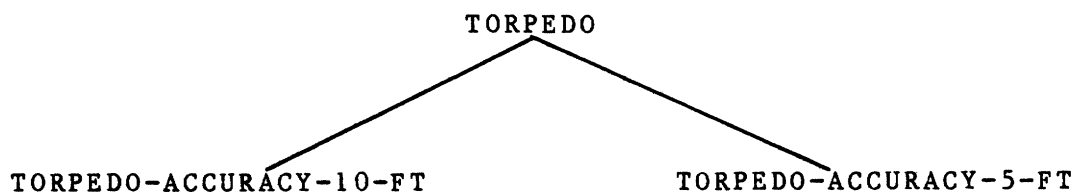


Figure 4.6 TORPEDO Sub-classes Based on Attribute ACCURACY

4.2.2 Selecting Salient DDAs -

The important attributes list also plays a major role in selecting the DDAs for a particular sub-class. Recall that the DDAs are a set of attributes which distinguish one sub-class from all other sub-classes in the same breakdown. They provide the generation system with salient descriptive information about the differences between the sub-classes. It is often the case that several sets of attributes distinguish a particular sub-class from the others (see section 5.3 for discussion of how these sets are found). In this situation, the important attributes list is consulted in order to choose the most salient distinguishing features. The set of attributes with the highest number of attributes

on the important attributes list is chosen.

A problem comes about when there is more than one set of potential DDAs having the highest number of attributes from the important attributes list. Since the important attributes list is not ordered by importance, there was no criteria for deciding among such sets. The only criteria for the set selected is that it should be small enough for the representation while transmitting the most salient features of the sub-class. The ties are divided into two cases. 1) a set of one attribute (1-set) is needed to distinguish the sub-class; 2) a set of more than one attribute is needed to distinguish the sub-class.

In handling case 1) the philosophy used is: since only 1-sets are being considered, the representation can afford to include several such sets. All 1-sets are included which are indistinguishable by means of the important attributes list. This can occur when either many of the 1-sets contain an attribute from the important attributes list or when none of the 1-sets contain an attribute from the important attributes list. We will consider each sub-case in turn. The first sub-case occurs when some of the 1-sets are made up of an attribute from the important attributes list. In this case all such sets are included since there is no way of determining which of these are better. An example of this is found in the DDAs for AIRCRAFT-CARRIER. Here 1-sets

World Knowledge Axioms

{LENGTH} and {DISPLACEMENT} distinguish an AIRCRAFT-CARRIER from other ship-types. Since both of these attributes appear on the important attributes list, both sets are included as the DDA.

The second sub-case occurs when none of the 1-sets include attributes from the important attributes list. In this case there is no basis for choosing one set over the others; so all sets are included in the DDA. An example of this is found in the breakdown of entity AIRCRAFT on the basis of its PROPULSION. The sub-class JET-AIRCRAFT has several 1-sets for its DDA. This is because several 1-sets distinguish the JET-AIRCRAFT from other AIRCRAFT, but none of these 1-sets appear on the important attributes list. The DDA for JET-AIRCRAFT include the following sets: {COMBAT-CEILING}, {MAXIMUM-CEILING}, {CRUISE-SPEED}, {MAXIMUM-SPEED}, and {FUEL-TYPE}.

Case 2) is the case where sets of more than one attribute are needed to distinguish the sub-class. In this case, since there are potentially many attributes in a set, only one set is included in the representation. Thus, ENHANCE chooses an arbitrary set from those containing the highest number of attributes from the important attributes list to be the DDA.

4.2.3 Specific Axiom Conclusions -

The specific axioms of the ENHANCE system take the form of a single list of attributes that are considered important to the domain. This list is termed the important attributes list and is used by ENHANCE in two major ways. First, ENHANCE attempts to form breakdowns based on character attributes in the list; if these breakdowns fail, then numeric attributes are used. Secondly, the important attributes list is used for generating the descriptive information associated with a sub-class. In particular, it is used to establish which set(s) of attributes should be used as the DDA for a sub-class when several such sets are available.

The important attributes list affords the user less control over the representation formed than the very specific axioms since it only suggests paths for the system to take. The system may attempt to form breakdowns based on attributes in the list, but these breakdowns will be subjected to more tests than breakdowns formed by the very specific axioms. (These tests are discussed in detail in section 4.3.3). The specific axioms (important attributes list) specify attributes that are, for one reason or another, important to the domain. Breakdowns based on these attributes are subjected to more tests since attributes important to the domain may not necessarily yield meaningful

World Knowledge Axioms

breakdowns. The very specific axioms, on the other hand, are not subjected to as many tests which eliminate the breakdowns since the breakdowns themselves were explicitly specified by the user. Thus the important attributes list gives the user less control over breakdowns formed. Ultimately the contents of the database dictates whether a breakdown will be included in the final representation.

4.3 General Axioms

The final type of world knowledge axioms used by ENHANCE are the general axioms. These axioms are domain independent and need not be changed by the user. They encode general principles used for deciding things like whether sub-classes formed should be added to the knowledge representation, and how sub-classes should be named.

These axioms are world knowledge even though they are not changed by a user of the system. They do make decisions that require outside knowledge. The type of knowledge that is depicted in these axioms is common to all database domains. Therefore, it is not necessary for the user to alter that knowledge.

One problem faced by a system which automatically generates sub-classes of the database entity classes, is naming the sub-classes. The name must uniquely identify a

sub-class and should give some semantic indication of the contents of the sub-class. These problems are handled by the general axioms entitled naming conventions.

A second problem that may occur with automatic sub-type generation is that some of the sub-classes in a particular breakdown may carry less meaning than others. For instance, some of the sub-classes may contain only one individual from the database. If several such sub-classes occur, then they are combined to form a CLASS-OTHER sub-class. This use of CLASS-OTHER compacts the representation while adding more meaning than the individual sub-classes did. For example, the DDA for CLASS-OTHER indicates what attributes are common to all entity instances that fail to make the criteria for membership in any of the larger named sub-classes. Without CLASS-OTHER, this information would have to be derived by the generation system; this is a potentially time consuming process. The general axioms include several rules which will block the formation of "CLASS-OTHER" in circumstances where it will not add information to the representation. These rules are discussed below.

Perhaps the most important use of the general axioms is their role in deciding if an entire breakdown adds meaning to the knowledge representation. A breakdown does not add meaning if its sub-classes simply rename the sub-classes of another breakdown. The general axioms also include rules

World Knowledge Axioms

for detecting and "filtering out" this type of breakdown.

4.3.1 Naming Conventions -

Naming the generated sub-classes is not an easy task for an automated system. The names should be unique, give semantic information about the contents of the sub-classes, and be reasonable to a natural language user of the ENHANCE system. In the case of breakdowns formed by the very specific axioms, the sub-class name is included as part of the axiom. In other cases, the sub-class name must be derived. ENHANCE handles the naming problem by making the sub-type name some combination of the database entity name along with the name and value of the attribute used to define the sub-class. (If the attribute used has a units field, the units will also be included in the sub-class name.)

Because of the components of the names, we are assured that the names are unique and will give some semantic indication of the contents of the sub-class. The semantic contents is indicated by the name and value of the attribute the breakdown is based on; the name of the entity class must be included to ensure the uniqueness of the sub-class name. Other rules must be used to insure that the names generated were reasonable in the natural language sense. The naming conventions used by ENHANCE are based on the type

of the attribute used to form the breakdown.

The first of these rules apply to character attributes whose name includes the word TYPE. In this case, the attribute value specifies the type; the attribute name specifies what the type refers to (i.e. type of what). In order to capture the semantic content both the value of the attribute and part of the attribute name must be included. For example, suppose that there are two different values in the database for attribute FUSE-TYPE of entity TORPEDO. Further suppose the values are TIMED and IMPACT. Combining the attribute name and value to form the sub-class names would result in TIMED-FUSE-TYPE-TORPEDO and IMPACT-FUSE-TYPE-TORPEDO. More natural names in this case would be: TIMED-FUSE-TORPEDO and IMPACT-FUSE-TORPEDO. Here, part of the attribute name must be included for semantic clarity; the entire attribute name is not necessary. In fact, using the entire attribute name adds unnecessary words to the sub-class name. The name formed will therefore be the concatenation of the defining attribute value followed by the part of the attribute name occurring before the word TYPE followed by the entity name. This is summarized in rule 1 below.

rule 1 -

The name of a sub-class of entity class ENT formed using a character attribute with a name of the form X-TYPE and value of VAL will be: VAL-X-ENT.

World Knowledge Axioms

The second rule applies to all other character attributes (i.e. those whose name does not include the word TYPE). In this case, ENHANCE does not include the attribute name in the name of the sub-class. The value of the attribute and the entity name carry enough semantic information. Including the attribute name makes the sub-class name rather cumbersome and awkward. The name of the parent entity, however, is needed for uniqueness reasons. Many entities in the ONR database have attribute FLAG indicating their country affiliation. Suppose that both a breakdown of SHIP and a breakdown of SUBMARINE were formed on the basis of attribute FLAG. If only the value of the attribute were used for the sub-class name, duplicate sub-class names would result. Therefore, the name formed is the concatenation of the defining attribute value and the parent entity name.

This rule is used for naming the sub-classes of SUBMARINE based on attribute CLASS. Since there are only two different values for attribute CLASS in the database (namely: WHISKY and ECHO-II), only two sub-classes are formed. Their names are WHISKY-SUBMARINE and ECHO-II-SUBMARINE. (Note that these names would be WHISKY-CLASS-SUBMARINE and ECHO-II-CLASS-SUBMARINE if they were made by simply combining the attribute value and name, and the parent entity name.)

rule 2 -

The name of a sub-class of entity ENT formed using a character attribute (whose name is not of the form X-TYPE) with value VAL will be: VAL-ENT.

Names must also be generated for sub-classes resulting from breakdowns based on numeric attributes. The numeric attributes are handled more uniformly than the character attributes, although there is some difference depending on whether a value for the units is specified. In this case, the sub-class name results from concatenating the entity name, the attribute name, the attribute value, and the value of the units field (if such a field was available). Examples of sub-class names formed in this way are: GUN-HORZ-RANGE-3900-YDS, MISSILE-LETHAL-RADIUS-200-FT, and MISSILE-MAXIMUM-ALTITUDE-4000-FT. These names, generated by rule 3, capture both the semantic contents of the sub-class and are appropriate from the natural language stand point.

rule 3 -

The name of a sub-class of entity ENT formed using a numeric attribute named ATT, with value NUMB, and an (optional) units value of UNITS will be: ENT-ATT-NUMB-UNITS.

4.3.1.1 Summary -

The general axioms for naming conventions consist of a set of rules for naming the sub-classes formed by various kinds of breakdowns. Since the values of different attributes carry varying degrees of semantic clues about

World Knowledge Axioms

their associated attribute name, the rules vary depending on the attribute the breakdown is based on. Numeric attribute values carry the least amount of semantic information. For this reason, the attribute name is always included in the sub-class name. Different character attributes carry varying amounts of semantic information. Attribute names of the form X-TYPE are combined with their values to form the sub-class name. Other character attribute values plainly indicate their corresponding name. Rather than making the sub-class name redundant, the attribute name is not included in the sub-class name for attributes of this type.

4.3.2 CLASS-OTHER Formation -

The CLASS-OTHER was originally conceived as a catch-all sub-class. It was to include 1) all individuals who did not fit into any of the sub-classes specified in a very specific axiom; 2) all individuals who fell into a sub-class by themselves. The idea behind 2) was that the system should not generate descriptive information for an individual since the information could be derived directly from the database. In practice, it was found that these ideas for CLASS-OTHER formation had some problems.

One such problem has to do with where breakdowns are attached in the generalization hierarchy. When more than one breakdown is made for a particular database entity, it is often the case that one breakdown is the refinement of another breakdown. In this case, it is desirable to attach the refinement breakdown under the other breakdown in the hierarchy. (See fitting algorithm in section 5.6 for explanation of how this is done.) Two such breakdowns are found for the entity SHIP in the ONR database. The breakdown based on attribute CLASS is a refinement of the breakdown based on SHIP-TYPE given in a very specific axiom (see Figure 4.1). For example, every SHIP that has CLASS = SKORY is of SHIP-TYPE = DESTROYER. In the final representation, it is more meaningful to attach the breakdown based on CLASS under the breakdown based on SHIP-TYPE rather than under SHIP itself. This desired attachment causes some problem for the CLASS-OTHER formation.

Suppose that there are several sub-types based on attribute CLASS with only one individual. This is actually the case in the ONR database used to test ENHANCE. Some such CLASSES are KITTY-HAWK and FORRESTAL (both refinements of SHIP-TYPE AIRCRAFT-CARRIER) along with KRIVAK, ADAMS-CF, and KOTLIN (all refinements of SHIP-TYPE DESTROYER). In the original formulation of CLASS-OTHER, these five sub-types

World Knowledge Axioms

would be combined to form the sub-type "OTHER-CLASS-SHIPS". This causes a problem since the breakdown based on CLASS is no longer a refinement of the breakdown based on SHIP-TYPE.

The above situation prompted the formation of the following general axiom for forming CLASS-OTHER:

rule 4 -
Combine only those sub-classes containing one individual into CLASS-OTHER that are refinements of the same superordinate. The CLASS-OTHER name should reflect that superordinate.

Applying this rule to the example above would prompt the formation of two CLASS-OTHER sub-types: OTHER-CLASS-AIRCRAFT-CARRIER and OTHER-CLASS-DESTROYER. These classes would enable the breakdown based on CLASS to be presented as a refinement of the breakdown based on SHIP-TYPE in the final representation.

Even when using this rule, several other problems arise. Using just rule 4 above, the breakdown shown in Figure 4.7 was generated.*

* The figure shows just a portion of the breakdown actually generated. Note: the entity name (SHIP) has been left off some of the sub-class names for reasons of space.

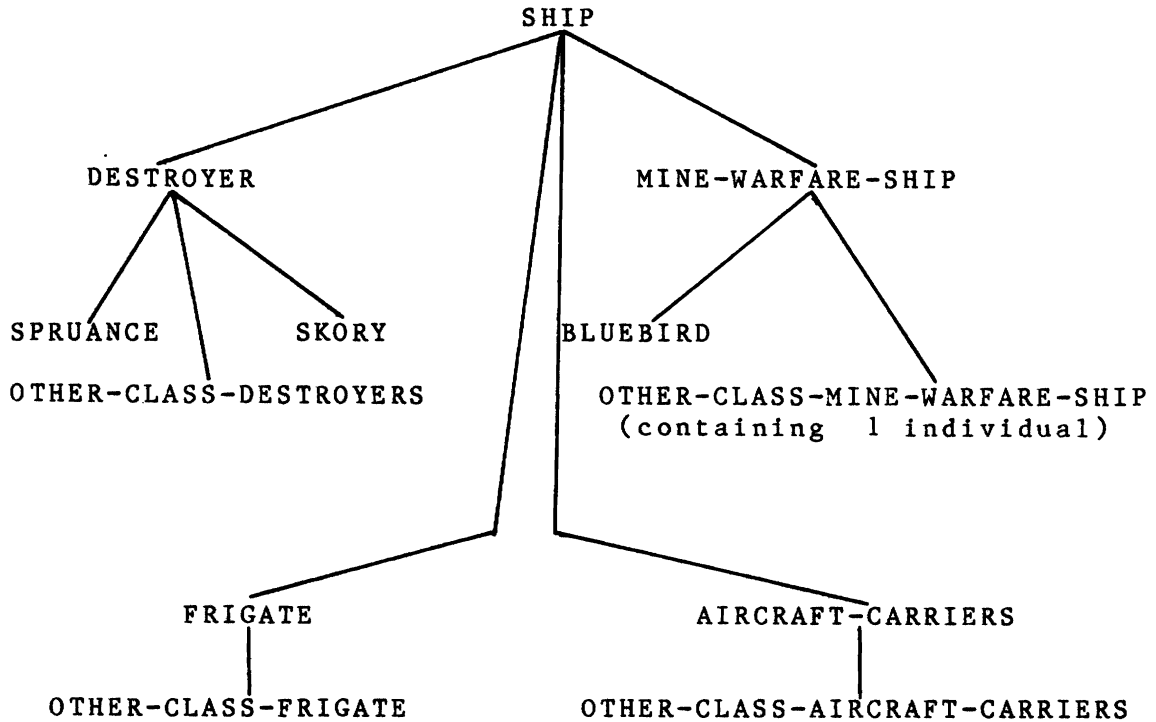


Figure 4.7 SHIP Breakdowns using CLASS-OTHER Rule 4.

One problem can be seen in the refinement of MINE-WARFARE-SHIP. Sub-class OTHER-CLASS-MINE-WARFARE-SHIP has been formed even though the CLASS-OTHER contains only one individual. In this case, there is no reason to form CLASS-OTHER. In fact, it would be more meaningful to leave the one individual in its own sub-class. That way, some of the characteristics of the individual ship will be reflected in its sub-type name. This observation led to CLASS-OTHER

World Knowledge Axioms

rule 5.

rule 5 -
Do not form CLASS-OTHER if it will contain
only one individual.

After applying rule 5 the MINE-WARFARE-SHIP would have the following sub-types: BLUEBIRD-SHIP and T-43-SHIP.

A second peculiarity that can be seen in the tree shown in Figure 4.7, is that the only sub-type of AIRCRAFT-CARRIER is OTHER-CLASS-AIRCRAFT-CARRIER (the FRIGATE has the same problem). This sub-type is odd for two reasons: first, the "OTHER" name leads one to believe that other sub-types of the AIRCRAFT-CARRIER exist; second, the one class shown exactly reflects the contents of the superordinate AIRCRAFT-CARRIER. To stop formation of CLASS-OTHER with these properties, a third CLASS-OTHER rule was implemented. It reads:

rule 6 -
Do not form CLASS-OTHER if it will be the
only child of a superordinate.

Using the above three rules, the tree structure shown in Figure 4.8 was generated. This structure, although a bit larger than that in Figure 4.7, carries more information.

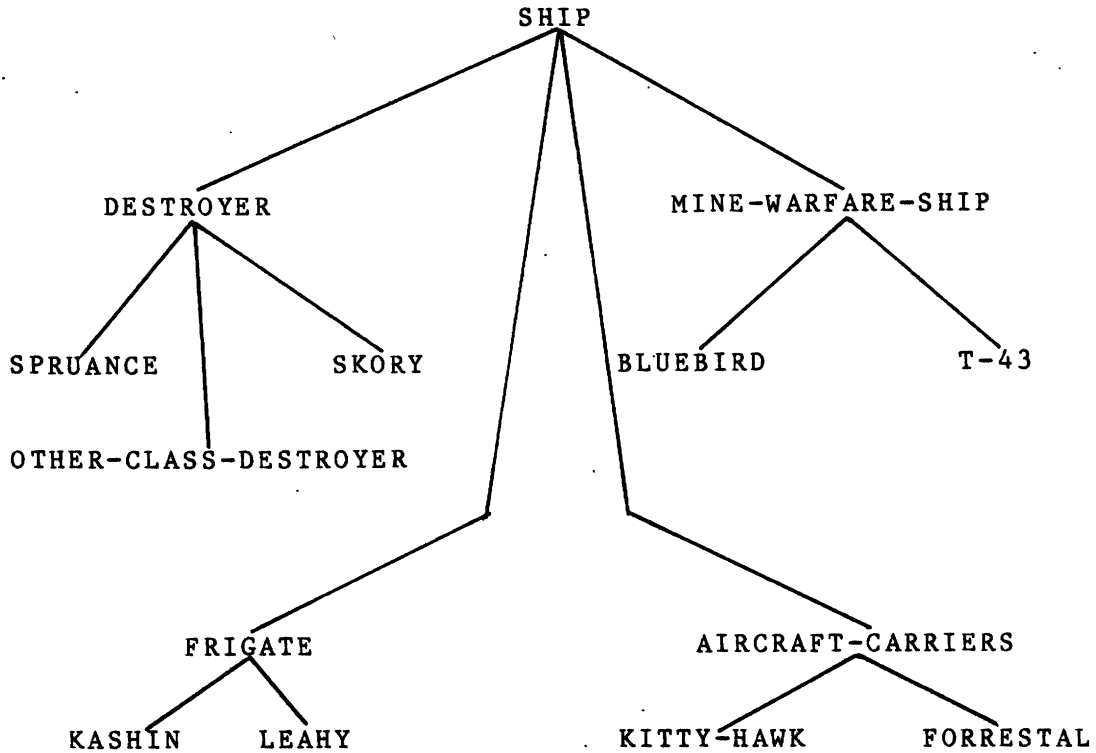


Figure 4.8 SHIP Breakdown Using All Three CLASS-OTHER Rules.

4.3.2.1 Summary -

The above examples illustrate the need for a group of general axioms dictating rules for the formation of CLASS-OTHER. CLASS-OTHER is a necessary item in an automatically generated representation. It serves to make the representation more concise while at the same time gives additional meaning to the representation. It was shown that in certain situations the CLASS-OTHER is not appropriate.

World Knowledge Axioms

The general axioms contain a set of rules for detecting such a situation. The rules developed for CLASS-OTHER formation yield a richer more appropriate structure.

4.3.3 Rules For Accepting A Breakdown -

A third major use of the general axioms is to decide which breakdowns are acceptable and which are not. Some of the breakdowns formed using the very specific and the specific axioms are not ones that add meaning to the knowledge representation. One role of the general axioms is to detect such breakdowns so that they will not be included in the final representation.

One kind of breakdown which fails to add information to the representation is a breakdown for which only one sub-type is formed. The sub-type, therefore, exhibits the same attributes as the entity class itself. This kind of breakdown occurs when every individual in the entity class has the same value for the feature defining the sub-class. In the ONR database used, this problem occurred in the entity class TORPEDO. Since FUSE_TYPE is on the important attributes list, a breakdown of TORPEDO based on attribute FUSE_TYPE was attempted. In the database used, every instance of a TORPEDO had FUSE_TYPE = IMPACT. Therefore, every instance fell into the same sub-class. This problem was detected by the general axioms and the breakdown based

on FUSE_TYPE was not included in the representation. The principle used is stated as general axiom rule 7 below.

rule 7 -

If a breakdown results in the formation of only one sub-type, then do not use that breakdown.

A similar problem occurs in a breakdown in which every sub-type contains only one instance. These sub-types exactly mirror the database instances. Clearly there would be no reason to have this information in the representation since it could be derived directly from the database itself. Because of the nature of the important attributes list, this situation could easily occur. Very often the primary key of an entity is considered an important attribute to the domain, and is therefore included in the important attributes list. This would, in turn, lead to a breakdown of the database entity class based on its primary key. If this were the case, then the breakdown would lead to a sub-class formed for each individual occurring in the database. To stop such breakdowns from being added to the knowledge representation, rule 8 was implemented.

rule 8 -

If a breakdown results in the formation of one sub-type for each instance in the database, then do not use that breakdown.

World Knowledge Axioms

A problem related to the two previous problems is that two different breakdowns may be formed which contain exactly the same database instances. Thus, the sub-types in one breakdown would be just a renaming of the sub-types in the other breakdown. Clearly it would not be necessary to use both breakdowns. Since they both indicate potentially the same information, one of these breakdowns is arbitrarily used in the representation. This problem is handled by rule 9.

rule 9 -

If two breakdowns contain exactly the same individuals, then use only one of them.

A breakdown may also not be useful if no DDAs can be found for one or more of its sub-classes. Recall that a DDA is a set of attributes whose value differentiates a particular sub-class from all other sub-classes in the breakdown (see Chpt 3). If there is at least one sub-class for which no DDA can be found, then the attribute that the sub-classes are based on must not exert any influence over the other attributes. By this I mean that the other attributes do not "cluster" according to the value of the attribute the breakdown is based on. In this case, the only difference between the sub-classes is the attribute the breakdown is based on. Thus, there is no real difference between the sub-classes in the breakdown and it is therefore not very useful.

World Knowledge Axioms

This is one place where the breakdowns made from the very specific axioms are treated differently than breakdowns based on the specific axioms. If the above situation occurs in the case of a breakdown based on a very specific axiom, the breakdown is accepted. If it occurs in the case of a breakdown based on a specific axiom, then the breakdown is thrown out. The reason for this is that it is assumed that a breakdown based on a very specific axiom is highly desired by the user. Since the system caters to the user, it ignores the fact that some sub-classes do not have any DDAs. (It should be noted that this case never occurred in the use of the ENHANCE system on the ONR database.)

In the case of breakdowns based on the specific axioms (important attributes list), it is assumed that the user never thought about a particular attribute as the basis of a breakdown. It is therefore necessary for the system to make extra checks to make sure the breakdowns formed are meaningful. If no DDAs can be found for a sub-class, it is assumed that the breakdown is not meaningful and the breakdown is therefore thrown out.

rule 10 -

If there is a sub-class in the breakdown for which no DDA can be found and if the breakdown is based on an attribute from the important attributes list (specific axioms), then that breakdown should not be used.

World Knowledge Axioms

The above rules point out some reasons for not accepting breakdowns formed using both the very specific and the specific axioms. The rules attempt to ensure that all breakdowns added to the knowledge representation add to the information included in the representation.

4.4 Conclusions

The ENHANCE system uses three types of world knowledge axioms. There are several uses for the axioms. The major goals of the axioms are to ensure that the breakdowns formed are meaningful and that the descriptive information used is appropriate. In addition, the axioms are a tool for the ENHANCE user to tune the representation to his/her particular needs.

The very specific axioms are dependent on the database itself. These are the most powerful axioms in terms of their effect on the final representation. These axioms enable the database manager to specify particular breakdowns to appear in the final representation. In order to form these axioms the user must have indepth knowledge of both the database itself and the domain the database reflects. They are provided for the proficient user who has predefined notions of what the representation should contain. For this reason, the very specific axioms are not required by the system. If the ENHANCE user has no preconceived breakdowns,

no very specific axioms need to be specified.

The second type of axioms are the specific axioms. These axioms take the form of a single list of attributes. The specific axioms are dependent on the domain the database reflects (rather than on the database itself). When ENHANCE is moved from one database to another database on the same domain, chances are that the specific axioms will remain basically unchanged. The list of attributes which make up the specific axioms is termed the important attributes list. This is simply a list of attributes which indicate commonly referred to features of the domain the database reflects. This list is used by ENHANCE in two major ways. First ENHANCE attempts to form breakdowns of database entity classes based on the attributes in the list. Secondly it uses attributes in the list as the descriptive information of a sub-class whenever possible. In this way, ENHANCE attempts to form the most salient breakdowns and descriptive information.

The final type of axioms are the very general axioms. These axioms are domain independent and are never changed by the user. They include general principles about naming conventions and sub-class formation, along with general rules for deciding if a breakdown will add meaning to the representation.

World Knowledge Axioms

Three types of axioms are used to make different kinds of decisions with varying amounts of information from the ENHANCE user. ENHANCE ensures that the knowledge representation will reflect the contents of the database; the world knowledge axioms are provided to ensure that the knowledge representation will meet the user's expectations.

5.0 DESCRIPTIVE INFORMATION (IMPLEMENTATION PRINCIPLES)

5.1 System Overview

The ENHANCE system consists of a set of independent modules; each is responsible for generating some piece of descriptive information for the sub-classes. When the system is invoked for a particular entity class, it first generates a number of breakdowns based on the values in the database. These breakdowns are passed from one module to the next and descriptive information is generated for each sub-class involved. This process is overseen by the general axioms which may throw out breakdowns for which pieces of the descriptive information can not be generated.

Before generating the breakdowns from the values in the database, the constraints on the values are checked and all units are converted to a common value. Any attribute values that fail to meet the constraints are noted in the representation and not used in the breakdown calculation. From the resulting values a number of breakdowns are generated using the very specific and specific axioms.

Implementation Principles

The breakdowns are first passed to the "fitting algorithm" (section 5.6). When two or more breakdowns are generated for an entity class, the sub-classes in one breakdown may be contained in the sub-classes of another. In this case, the sub-classes in the first breakdown should appear as the children of the sub-classes of the second breakdown. The fitting algorithm is used to calculate where the sub-classes fit in the generalization hierarchy. After the fitting algorithm is run, the general axioms may intervene to throw out any breakdowns which are essentially duplicates of other breakdowns (see rule 9 above).

At this point, the DDAs of the sub-classes within each breakdown are calculated. The algorithm used in this calculation is given in section 5.3. If no DDAs can be found for a breakdown formed using the important attributes list, the general axioms may again intervene to throw out that breakdown (rule 10 above).

Next the system goes through a number of modules responsible for calculating the based DB attribute and for recording constant DB attributes and relation attributes. The actual nodes are then generated and added to the hierarchy. At this point any constant DB attribute values are propagated up the hierarchy as far as possible.

Implementation Principles

The calculation of some of the descriptive information involves many combinatoric problems. Some considerations taken to avoid these problems are discussed below.

5.2 Based DB Attribute

The based DB attribute of a sub-class is the attribute whose value defines the sub-class; it is the attribute and associated value within a sub-class that the partition is based on. The based DB attribute is used by the generation system to identify why a particular individual is a member of one sub-class as opposed to another.

In the case of a breakdown based on an attribute from the important attributes list, the based DB attribute for a sub-class is simply that attribute the breakdown is based on along with its associated value (see section 4.2.1 for a discussion of how this attribute is chosen). For example, in the ONR database used for the implementation, a breakdown of AIRCRAFT is made based on attribute PROPULSION. Since there are three values of PROPULSION in the database, the following sub-classes are formed: JET-AIRCRAFT, PROP-AIRCRAFT, and ZPROP-AIRCRAFT. The corresponding based DB attributes are (PROPULSION = JET), (PROPULSION = PROP), and (PROPULSION = ZPROP).

Implementation Principles

For breakdowns formed using the very specific axioms, the based DB attribute may take the form of a disjunction and/or specify a partial field (in the case of character attributes) or a range of values (in the case of numeric attributes) as the defining value. A disjunction is used when there are several reasons for placing an individual in a particular sub-class (i.e. when two or more different values can be used to place an individual in the same sub-class). The based DB attribute is calculated using the very specific axiom itself. For each sub-class formed, the based DB attribute is the disjunction of all of the ways, specified in the axiom, that an instance can be a member of the sub-class. An example of this is found in the sub-class of SHIP denoted CRUISER. (See figure 4.1 for the very specific axiom associated with the breakdown.) As specified in the axiom, a CRUISER is a SHIP whose first two characters of the HULL_NO are either CA or CG or CL. Therefore, the based DB attribute of CRUISER is (HULL_NO (1 2 CA) (1 2 CG) (1 2 CL)). In this case the based DB attribute is read, characters 1 through 2 of the HULL_NO are equal to CA or the characters 1 through 2 are equal to CG or characters 1 through 2 are equal to CL. Thus, an individual ship can be identified as being in the sub-class CRUISER in three different ways.

The based DB attribute of a sub-class may also be in the form of a disjunction due to class-other formation (see section 4.3.2). Class-other is the result of combining together several sub-classes which originally contain only one member. In this case, the based DB attribute of the class-other would be the disjunction of the individual based DB attributes from each of the combined sub-classes.

5.3 Distinguishing Descriptive Attributes (DDAs)

The Distinguishing Descriptive Attributes (DDAs) of a sub-class is a set of attributes, other than the based DB attribute, whose collective value differentiates that sub-class from all other sub-classes in the same breakdown. The DDA exhibits some salient distinction between the given sub-class and all others. It can be used by the generation system to explain the difference between two sub-classes in the same breakdown.

Finding the DDA of a sub-class is a problem which is combinatoric in nature since it may require looking at all combinations of the attributes of the entity class. This problem is accentuated since it has been found that in practice, a set of attributes which differentiates one sub-class from all other sub-classes in the same breakdown does not always exist. Unless this problem is identified ahead of time, the system would go through all combinations

Implementation Principles

of all of the attributes before deciding the sub-class can not be distinguished. Since it is often the case that a sub-class can not be completely distinguished from all others, it was necessary to form some guidelines concerning exactly what constitutes a DDA and how to go about finding it.

There are several features of the set of DDAs which are desirable. 1) The set should be as small as possible. That is, it should contain only enough attributes to distinguish the sub-class and no more. The reason for this is to keep the knowledge representation as concise as possible. It should capture as much information as possible in the least amount of space. 2) The set of DDAs should be made up of salient attributes (where possible). Suppose that a set of two attributes will distinguish a sub-class from all others. There may be several different combinations of two attributes that serve this purpose. In this case, that set containing the most attributes from the important attributes list is chosen. In this way, the DDA chosen is as meaningful to the user as possible. 3) The set of DDAs of a sub-class should add information about that sub-class not already derivable from the representation. That is, the DDA should include attributes that make the sub-class important in its own right; therefore the attributes chosen should be different from DDAs of the parent sub-class. Figure 5.1

shows a small portion of the breakdown generated for entity class SHIP.

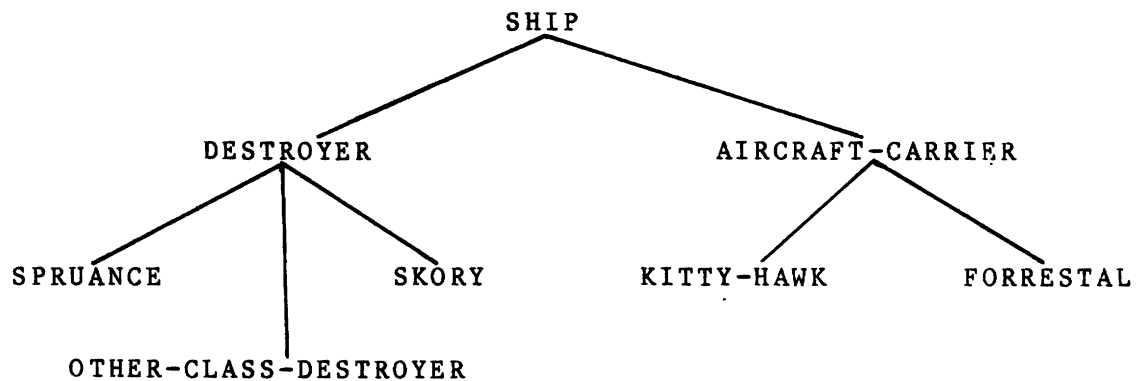


Figure 5.1 Portion of breakdown of entity class SHIP

The DDA calculated for sub-class DESTROYER was DRAFT = 15 - 222. Thus, the DRAFT of a DESTROYER is identified as being an important attribute for distinguishing the sub-class. As seen in the figure, the SKORY is a sub-class of the DESTROYER. It therefore inherits all of the aspects of the DESTROYER. Thus, the DRAFT of the SKORY is identified as being one important distinguishing feature. Even though the SKORY, which has a DRAFT = 15, could be distinguished from all other classes of SHIP by its DRAFT, using the DRAFT as the DDA would be somewhat redundant. It would be more meaningful to identify other attributes which could be used to distinguish the SKORY. In the ENHANCE system, therefore, DDAs of the parent sub-class are not considered in the

Implementation Principles

calculation of the DDA for the child sub-class.

In generating the DDA for a sub-class, ENHANCE takes several steps to ensure both that the DDA has the above desirable features and that the combinatoric problems identified are avoided. A brief outline of the method used by ENHANCE is given along with justification for some of its decisions.

In order to calculate the DDAs of a given sub-class, ENHANCE must have some way of comparing the attribute values within the sub-class with the attribute values for other sibling sub-classes. For this purpose, ENHANCE generates a list containing 1) the maximum and minimum values of all numeric attributes and 2) any constant values for all character attributes for each sub-class in the breakdown. This list is used to make comparisons between the sub-classes.

Once the means for comparing the sub-classes had been established, the method for generating the DDAs was originally thought to be evident. The system could simply start generating all 1-combinations of attributes, followed by 2-combinations etc.. until a set of attributes was found which differentiated the sub-class. To insure that the DDA was made up of the most meaningful attributes, combinations of attributes from the important attributes list could be

generated first.

This method, although conceptually clear, was not very practical. It is often the case that some of the attributes of the sub-class never differentiate the sub-class from any others. Using these attributes in the combinations above would be of no use. It is also the case that some attributes can be identified as the only means of differentiating the sub-class from some other sub-class. Therefore, any combination of attributes not including those attributes would fail to differentiate the sub-class. Identifying these two types of attributes before the combinations of attributes are formed, cuts down on much of the time spent forming the DDA.

For these reasons, a "pre-processor" to the combination stage of the calculation was developed. The combinations are formed of only Potential-DDAs. These are a set of attributes whose value can be used to differentiate the sub-class from at least one other sub-class. That is, the attributes included in potential-DDAs take on a value within the sub-class that is different from the value the attributes take on in at least one other sub-class. Using the potential-DDAs ensures that each attribute in a given combination is useful in distinguishing the sub-class from the others.

Implementation Principles

Calculating the potential-DDAs requires comparing the values of the attributes within the sub-class with the values within each other sub-class in turn. If, for a particular sub-class, this comparison yields only one attribute, then this attribute is the only means for differentiating that sub-class from the one the DDAs are being calculated for. Thus, the DDA must contain that attribute. Attributes of this type are called definite-DDAs.

The system uses the potential-DDAs and the definite-DDAs to find the smallest and most salient set of attributes to use as the DDA. It first checks to see if the definite-DDAs alone are enough to differentiate the sub-class. If so, they are selected as the DDA. Otherwise, ENHANCE tries to differentiate the sub-class using the definite-DDAs and one attribute from the potential-DDAs. If this fails, it attempts using two attributes from the potential-DDAs, and so forth.

When a set of a attributes of a particular length is found to differentiate the sub-class, it is usually the case that many sets exist. If so, ENHANCE uses the important attributes list to select the set of attributes containing the most salient attributes. (See section 4.2.2 for a discussion of the issues involved in choosing this set.) In this way, the DDA is calculated to be the smallest most

Implementation Principles

salient list of attributes whose collective value differentiates the sub-class from all others in the breakdown.

The above description does not take into account the possible inability to distinguish a sub-class from all other sub-classes. The inability to distinguish the sub-class from another is very often due to the value of a particular attribute within the sub-class overlapping that of another sub-class by some small amount. An example of this is seen in the ONR database for the AIRCRAFT-CARRIER. The DISPLACEMENT of the AIRCRAFT-CARRIER could serve as a DDA except that it overlaps the DISPLACEMENT of the FRIGATE. The DISPLACEMENT of the AIRCRAFT-CARRIER = 78000-80800, the DISPLACEMENT of the FRIGATE = 5200-7800. Here the only overlap occurs at the endpoints; the ranges themselves are actually quite different. It was decided that where attribute values overlap by such a small amount, they can be said to distinguish the sub-class. In the implementation 15% overlap was permitted. This number may be changed for different domains and even for different databases. For this reason, any implementation should make this number very accessible so that it can be easily changed when necessary.

Implementation Principles

Even when allowing this small amount of overlap, it was common not to be able to differentiate a given sub-class from other sub-classes in the breakdown. If this number of sub-classes is small (compared to the total number of sub-classes in the breakdown) they are disregarded for the DDA calculation. In such a case, the DDA is marked in the knowledge representation to indicate the sub-classes which fail to be differentiated. If, on the other hand, the number of such sub-classes is higher than some predetermined percentage of the total number of sub-classes, then the system concludes that no DDA can be found for the sub-class. This predetermined percentage is another aspect of the implementation which should be made accessible to the user so that it can be changed from one database to another.

Using the potential-DDAs, definite-DDAs and allowable overlaps reduces much of the time spent in the DDA calculation.

5.4 Constant DB Attributes

Since the sub-classes formed by the ENHANCE system inherit all of the attributes of the database entity class, no new attributes are attached to the sub-class. The sub-class does, however, restrict the values that a given attribute takes on. In some cases an attribute may take on a constant value over a sub-class. Such information is

Implementation Principles

beneficial for the generation system to have and is therefore recorded by ENHANCE in the DB attributes list of a sub-class. In order to record this information, ENHANCE must look at the value of every attribute for every instance which falls into a given sub-class. This, however, is necessary in the calculation of the DDA. The list containing the minimum and maximum numeric attribute values and constant character attribute values used in the DDA calculation, is used to record values of certain attributes in the DB attributes list of the sub-class.

There are two cases when values are recorded in the DB attributes list. 1) all attributes with a constant value over the sub-class are recorded. 2) all attributes that are used in the DDA for sibling sub-classes are recorded. In this case, the value of the attribute may not be constant, instead it may be a range of values.

This additional information in the DB attributes list allows the generation system to do comparisons between sibling sub-classes. The DDAs of a sub-class are not really meaningful unless the attribute values that make up the DDA can be compared with the values of the same attributes for other sub-classes. This additional information allows the generation system to calculate the relationship between the attributes in the DDA of one sub-class to all other sub-classes. This leads to statements from the generation

Implementation Principles

system like: the AIRCRAFT-CARRIER has a larger DISPLACEMENT than most other SHIPS.

5.5 Constant Relation Attributes

The values of attributes associated with relations may also be restricted within a sub-class. The range of values that these relation attributes take on are also recorded by the ENHANCE system. This requires looking up in the database, the value of the relation attributes for each instance in the sub-class. This information gives the generation system one more means of comparing two entities that participate in the same relation.

5.6 Subset Entities List

The subset entities list of a node contains the immediate descendants of that node in the generalization hierarchy. These children are grouped into mutually exclusive sets. The subset entities list is used by the generation system to give a definition of a particular object in terms of its constituents. It allows the generation system to say things like: There are two kinds of submarines in the ONR database, Whisky-submarines and Echo-II-submarines.

Implementation Principles

When two or more breakdowns are formed for a given entity class, the sub-classes in one breakdown may be contained in the sub-classes of another breakdown. In this case the sub-classes of the first breakdown are actually the children of the sub-classes of the second breakdown. They therefore should appear in the subset entities list of the sub-classes of the second breakdown rather than in the subset entities list of the entity class itself. The sub-classes in the second breakdown are termed the parents of the sub-classes in the first breakdown. In this situation the first breakdown is said to fit under the second. Deciding which breakdowns fit under each other is another problem which is combinatoric in nature. It is decided by the "fitting algorithm" which is briefly described here.

In order to decide what breakdowns fit under each other, there must be a way to compare the database instances falling into each sub-class. For this reason, a list is generated for each breakdown which contains the primary keys of each individual in each sub-class in the breakdown. This is called the individuals list. One breakdown is said to fit under another if all of the individuals contained in each sub-class in the first breakdown are contained within some sub-class of the second breakdown.

Implementation Principles

To illustrate what is involved in determining where each breakdown fits into the hierarchy, consider the following example. Figure 5.2 is a hypothetical example of an individuals list containing four breakdowns.

```
((b1 (s1 i1 i2 i3) (s2 i4 i5 i6))
 (b2 (s3 i1 i3 i5) (s4 i2 i4 i6))
 (b3 (s5 i1 i2) (s6 i3) (s7 i4) (s8 i5 i6))
 (b4 (s9 i1) (s10 i2) (s11 i3) (s12 i4) (s13 i5) (s14 i6)))
```

Figure 5.2 Hypothetical Individuals List

In the figure b1, b2, b3, and b4 are the names of the four breakdowns. s1 and s2 are the sub-classes contained in b1. s3 and s4 are the sub-classes contained in b2. etc.. s1 contains three individuals whose primary keys are i1, i2, and i3.

We see that b1 and b2 do not fit under any breakdowns since the sub-classes in these two breakdowns are not contained within the sub-classes of another breakdown. (In this case the parent of b1 and b2 will be the entity class itself. Therefore they will appear as two mutually exclusive sets in the subset entities list of the entity class.) b3 fits under b1 since each sub-class in b3 is contained in a sub-class in b1. (The individuals in s5 and s6 are contained in s1, the individuals in s7 and s8 are

Implementation Principles

contained in s2.) b4 fits under both b1 and b3. In order to calculate the parent breakdown of b4, ENHANCE finds the least general breakdown that b4 fits under. Since b4 fits under both b1 and b3, and b3 fits under b1, the least general parent of b4 is b3. Therefore, the sub-classes of b4 will be contained in the subset entities list of b3.

The tree structure generated for the individuals list in Figure 5.2 is shown in Figure 5.3. Mutual exclusion is indicated by a line joining the arcs pointing to mutually exclusive subsets.

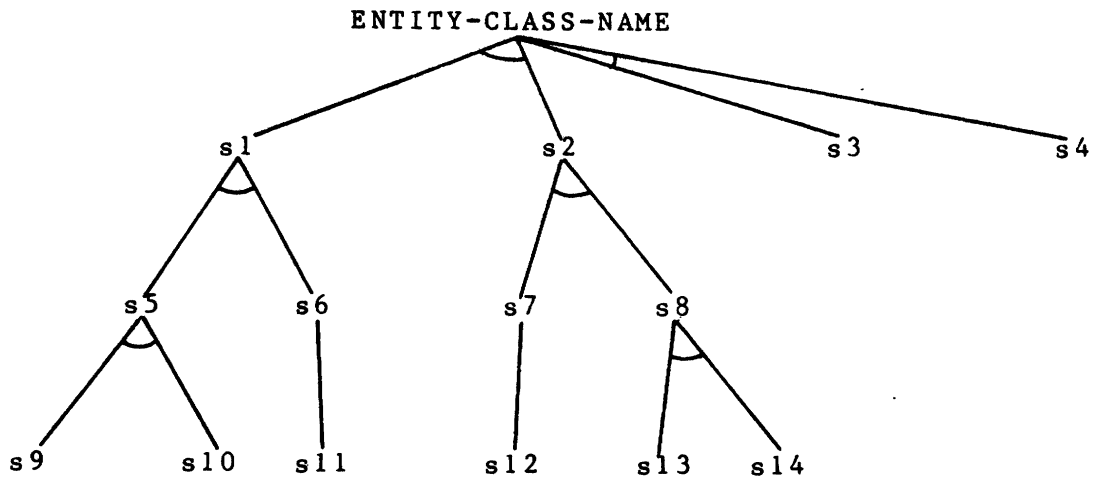


Figure 5.3 Tree Structure Generated for Figure 5.2.

Uses

6.0 USE OF KNOWLEDGE REPRESENTATION GENERATED BY ENHANCE

The following examples illustrate how the TEXT system uses the information generated by ENHANCE. The examples are taken from actual output generated by TEXT. Following each example is a brief description of where each piece of information was found in the representation.*

Example 1

Question: What do you know about ECHO-II-SUBMARINES?

Answer:

1) Echo IIs have a PROPULSION_TYPE of NUCL and a FLAG of RDRD. 2) All echo IIs in the ONR database have REMARKS of 0, FUEL_TYPE of NUCL, IRCS of 0, MAXIMUM_OPERATING_DEPTH of 700 and NORMAL_OPERATING_DEPTH of 100. 3) There are no sub-classes of echo II in the ONR database. 4) Echo IIs carry 16 torpedoes, between 16 and 99 missiles and 0 guns. 5) A submarine is classified as an echo II if its CLASS is ECHO II.

The distinguishing descriptive attributes (DDAs) of the ECHO-II are first used to identify its major features (sentence 1). The DB attributes list is then consulted and

* The sentences are numbered here to simplify the discussion: there are no sentence numbers in the actual material generated by TEXT.

the constant attributes of the ECHO-II are identified (sentence 2). The third sentence is derived from the subset entities list while the fourth comes from the recorded values of the relation attributes associated with the ECHO-II. The final sentence explains the reason for classifying a particular submarine as an ECHO-II. This information is found using the based DB attribute of the ECHO-II.

Example 2

Question: What is an AIRCRAFT-CARRIER?

Answer:

1) An aircraft carrier is a surface ship with a DISPLACEMENT between 78000 and 80800 and a LENGTH between 1039 and 1063. 2) Aircraft carriers have a greater LENGTH than all other ships and a greater DISPLACEMENT than most other ships. 3) Mine warfare ships, for example, have a DISPLACEMENT of 320 and a LENGTH of 144. 4) All aircraft carriers in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL, BEAM of 252, ENDURANCE_RANGE of 4000, ECONOMIC SPEED of 12, ENDURANCE_SPEED of 30 and PROPULSION of STMTURGRD. 5) A ship is classified as an aircraft carrier if the characters 1 through 2 of its HULL_NO are CV.

In this example, the distinguishing descriptive attributes of AIRCRAFT-CARRIER are used to identify its features (sentence 1) and to make a comparison between AIRCRAFT-CARRIERS and all other types of ships (sentences 2

Uses

and 3). Since the ENHANCE system ensures that the values of the DDA for one sub-class appear in the DB Attribute list of every other sub-class in the same breakdown, the comparisons between the sub-classes are easier. Moreover, since ENHANCE has selected out several attributes as more important than others (based on the world knowledge axioms), TEXT can make a meaningful comparison instead of one less relevant. The final sentence is derived from the Based DB Attribute of AIRCRAFT-CARRIER.

Example 3

Question: What is the difference between an OCEAN-ESCORT and a CRUISER?

Answer:

1) The cruiser and the ocean escort are surface ships. 2) The ship's surface-going capabilities are provided by the DB attributes DISPLACEMENT and DRAFT. 3) The ship has DB attributes MAXIMUM SPEED, PROPULSION, FUEL(FUEL CAPACITY and FUEL TYPE), DIMENSIONS, SPEED DEPENDENT RANGE and OFFICIAL_NAME. 4) Ocean escorts have a DISPLACEMENT between 3400 and 4100. 5) All ocean escorts in the ONR database have REMARKS of 0, FUEL_TYPE of BNKR, FLAG of BLBL and MAST_HEIGHT of 85. 6) Ocean escorts carry between 2 and 22 torpedoes, 16 missiles and between 1 and 2 guns. 7) A ship is classified as an ocean escort if the characters 1 through 2 of its HULL_NO are DE. 8) Cruisers have a PROPULSION of STMTURGRD and a LENGTH between 510 and 673. 9) All cruisers in the ONR database have REMARKS of 0. 10) Cruisers carry between 8 and 42 torpedoes, between 4 and 98 missiles and between 1 and 4 guns. 11) A ship is classified as a cruiser if the characters 1 through 2 of its HULL_NO are CL or the characters

1 through 2 of its HULL_NO are CG. 12) The ocean escort, therefore, has a smaller LENGTH and a smaller DISPLACEMENT than the cruiser.

The first sentence is derived from the fact that both Ocean-Escorts and Cruisers are sub-types of entity class SHIP. TEXT then goes on to describe some characteristics of the SHIP (sentences 2 and 3). Information about the ship is part of the hand coded representation, it is not generated by ENHANCE. Next, the distinguishing features of the Ocean-Escort are identified followed by a listing of its constant attributes (sentences 4 and 5). The values of the relation attributes are then identified (sentence 6) followed by a statement based on the based bd attribute of the Ocean-Escort. Next this same type of information is used to generate parallel information about the Cruiser. The text closes with an inference based on the DDAs of the two types of ships.

Example 4

Question: What is a WHISKY-SUBMARINE?

Answer:

1) A whisky is an underwater submarine with a PROPULSION_TYPE of DIESEL and a FLAG of RDOR. 2) A submarine is classified as a whisky if its CLASS is WHISKY. 3) All whiskies in the ONR database have REMARKS of 0, FUEL_CAPACITY of 200, FUEL_TYPE of DIESEL, IRCS of 0, MAXIMUM_OPERATING_DEPTH of

Uses

700, NORMAL_OPERATING_DEPTH of 100 and
MAXIMUM_SUBMERGED_SPEED of 15.

In this example the distinguishing features of the Whisky Submarine are first identified (sentence 1). The based DB attribute of the Whisky is then used to explain how a submarine is identified as a Whisky (sentence 2). The third sentence identifies the constant database attributes associated with the Whisky-Submarine.

7.0 FUTURE WORK

There are several extensions of the ENHANCE system which would make the knowledge representation more closely reflect the real world. These include (1) the use of very specific axioms in the calculation of descriptive information and (2) the use of relational information as the basis for a breakdown.

At the present time, all descriptive sub-class information is calculated from the actual contents of the database, although sub-class formation may be based on the very specific axioms. The database contents may not adequately capture the real world distinctions between the sub-classes. For this reason, a set of very specific axioms specifying descriptive information could be adopted. The need for such axioms can best be seen in the DDA generated for ship sub-type AIRCRAFT-CARRIER. Since there are no attributes in the database indicating the function of a ship, there is no way of using the fact that the function of an AIRCRAFT-CARRIER is to carry aircraft, to distinguish AIRCRAFT-CARRIERS from other ships. This is, however, a very important real world distinction. Very specific axioms could be developed to allow the user to specify these important distinctions not captured the the contents of the

Future Work

database.

The ENHANCE system could also be improved by utilizing the relational information when creating the breakdowns. For example, missiles can be divided into sub-classes on the basis of what kind of vehicles they are carried by. AIR-TO-AIR and AIR-TO-SURFACE missiles are carried on aircraft, while SURFACE-TO-SURFACE missiles are carried on ships. Thus, the relations often contain important sub-class distinctions that could be used by the system.

8.0 CONCLUSION

A system has been described which automatically creates part of a knowledge representation used for natural language generation. Sub-type information is created for the database entity classes based on the contents of the database. ENHANCE uses particular values in the database to divide an entity class into a number of sub-classes. Descriptive information is created for these sub-classes using the remaining database values. Many problems which must be considered in this calculation have been identified and solutions proposed.

The contents of the database alone, however, are not enough to ensure a meaningful representation. Several ways in which an automatically generated representation may deviate from a user's expectation have been anticipated. A set of world knowledge axioms is employed to ensure that the representation formed by ENHANCE meets the expectations of the user.

Automatically generating part of the knowledge representation saves the database designer the tedious task of creating the entire knowledge representation by hand. (The coding of the knowledge representation is often

Conclusion

considered a bottleneck to the portability of the generation system.) Using the contents of the database along with the world knowledge axioms ensures that the representation reflects both the database itself and a user's view of the database. This ensures a consistent view of the database. At the same time, the world knowledge axioms provide the database designer with the means of tuning the representation to her/his needs.

The ENHANCE system also provides the generation system with a richer description of the database structure. This enables the generation of richer text without the use of extensive sub-type inferencing.

9.0 REFERENCES

[Blackman 73]. Blackman, R.V. (Ed.), Jane's Fighting Ships 1972-73, McGraw-Hill, 1973.

[Carrison 68]. Carison, D.J., The United States Navy, Frederick A. Praeger - Publisher, 1968.

[Chen 76]. Chen, P.P.S., "The Entity-Relationship Model - Towards a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No. 1, 1976.

[Grosz et. al. 82]. Grosz, B., et. al., "TEAM: A Transportable Natural Language System", Tech Note 263, Artificial Intelligence Center, SRI International, Menlo Park, Ca., (to appear).

[Lee & Gerritsen 78]. Lee, R.M., and Gerritsen, R., "Extended Semantics for Generalization Hierarchies", Proceedings of the 1978 ACM-SIGMOD International Conference on Management of Data, Austin, Texas, May 31 to June 2, 1978.

[Malhotra 75]. Malhotra, A., Design criteria for a knowledge-based English language system for management: an experimental analysis. MAC TR-146, MIT, Cambridge, Mass. (1975).

[McKeown 82]. McKeown, K.R., "Generating Natural Language Text in Response to Questions About Database Structure", Ph.D. Dissertation, University of Pennsylvania, Philadelphia, Pa., 1982.

[Smith & Smith 77]. Smith, J.M., and Smith, D.C.P., "Database Abstractions: Aggregation and Generalization", ACM Transactions on Database Systems, Vol. 2, No. 2, June 1977.

[Palmer 1975]. Palmer, J. (compiled by), Jane's Dictionary of Naval Terms, Macdonald & Jane's, London, 1975.

[Tennant 79]. Tennant, H., "Experience with the evaluation of natural language question answerers." Working paper #18, Univ. of Illinois, Urbana-Champaign, Ill. (1979).

APPENDIX A

LIST OF ATTRIBUTES ASSOCIATED WITH EACH ENTITY

SHIP

("OFFICIAL_SHIP_NAME" "char")
("SHIP_HULL_NO" "char" t)
("SHIP_CLASS" "char")
("SHIP_FLAG" "char")
("SHIP_PROPULSION" "char")
("SHIP_IRCS" "char")
("SHIP_FUEL_TYPE" "char")
("SHIP_FUEL_CAPACITY" "num")
("MAXIMUM_SHIP_SPEED" "num")
("ENDURANCE_SHIP_SPEED" "num")
("ECONOMIC_SHIP_SPEED" "num")
("ENDURANCE_SHIP_RANGE" "num")
("ECONOMIC_SHIP_RANGE" "num")
("SHIP_LENGTH" "num")
("SHIP_BEAM" "num")
("SHIP_DRAFT" "num")
("SHIP_DISPLACEMENT" "num")
("SHIP_MAST_HEIGHT" "num")
("SHIP_REMARKS" "char")

SUBMARINE

("OFFICIAL_SUB_NAME" "char")
("SUB_HULL_NO" "char" t)
("SUB_CLASS" "char")

LIST OF ATTRIBUTES ASSOCIATED WITH EACH ENTITY

("SUB_FLAG" "char")
("SUB_IRCS" "char")
("SUB_PROPULSION_TYPE" "char")
("SUB_FUEL_TYPE" "char")
("SUB_FUEL_CAPACITY" "num")
("SUB_MAXIMUM_SUBMERGED_SPEED" "num")
("SUB_MAXIMUM_OPERATING_DEPTH" "num")
("SUB_NORMAL_OPERATING_DEPTH" "num")
("SUB_REMARKS" "char")

AIRCRAFT

("AIRCRAFT_NAME" "char" t)
("AIRCRAFT_PRIMARY_ROLE" "char")
("AIRCRAFT_DESCRIPTION" "char")
("AIRCRAFT_FLAG" "char")
("AIRCRAFT_PROPULSION" "char")
("AIRCRAFT_FUEL_TYPE" "char")
("AIRCRAFT_FUEL_CAPACITY" "num")
("AIRCRAFT_REFUEL_CAPABILITY" "char")
("AIRCRAFT_MAXIMUM_SPEED" "num")
("AIRCRAFT_CRUISE_SPEED" "num")
("AIRCRAFT_MAXIMUM_CEILING" "num")
("AIRCRAFT_COMBAT_CEILING" "num")
("AIRCRAFT_COMBAT_RADIUS" "num")
("AIRCRAFT_CRUISE_RADIUS" "num")
("AIRCRAFT_REMARKS" "char")

GUN

("GUN_NAME" "char" t)
("GUN_DESCRIPTION" "char")
("GUN_HORZ_RANGE" "num")
("GUN_HORZ_RANGE_UNITS" "char")
("GUN_VERT_RANGE" "num")
("GUN_VERT_RANGE_UNITS" "char")
("GUN_ACCURACY" "num")
("GUN_ACCURACY_UNITS" "char")
("GUN_FIRE_RATE" "num")
("GUN_FIRE_RATE_UNITS" "char")
("GUN_REMARKS" "char")

LIST OF ATTRIBUTES ASSOCIATED WITH EACH ENTITY

MISSILE

("MISSILE_NAME" "char" t)
("MISSILE_SPEED" "num")
("MISSILE_DESCRIPTION" "char")
("MISSILE_MAXIMUM_ALTITUDE" "num")
("MISSILE_MAXIMUM_ALTITUDE_UNITS" "char")
("MISSILE_MINIMUM_ALTITUDE" "num")
("MISSILE_MINIMUM_ALTITUDE_UNITS" "char")
("MISSILE_HORZ_RANGE" "num")
("MISSILE_HORZ_RANGE_UNITS" "char")
("MISSILE_TIME_TO_TARGET" "num")
("MISSILE_TIME_TO_TARGET_UNITS" "char")
("MISSILE_LETHAL_RADIUS" "num")
("MISSILE_LETHAL_RADIUS_UNITS" "char")
("MISSILE_PROBABILITY_OF_KILL" "real")
("MISSILE_REMARKS" "char")

BOMB

("BOMB_NAME" "char" t)
("BOMB_DESCRIPTION" "char")
("BOMB_LETHAL_RADIUS" "num")
("BOMB_LETHAL_RADIUS_UNITS" "char")
("BOMB_WEIGHT" "num")
("BOMB_WEIGHT_UNITS" "char")
("BOMB_REMARKS" "char")

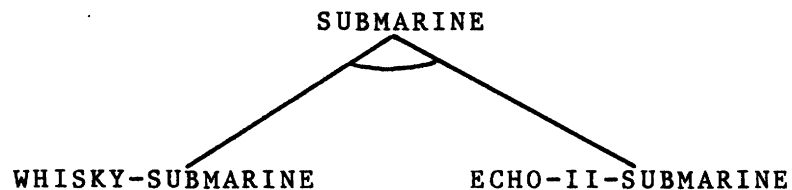
TORPEDO

("TORPEDO_NAME" "char" t)
("TORPEDO_FUSE_TYPE" "char")
("TORPEDO_DESCRIPTION" "char")
("TORPEDO_MAXIMUM_DEPTH" "num")
("TORPEDO_HORZ_RANGE" "num")
("TORPEDO_HORZ_RANGE_UNITS" "char")
("TORPEDO_ACCURACY" "num")
("TORPEDO_ACCURACY_UNITS" "char")
("TORPEDO_TIME_TO_TARGET" "num")
("TORPEDO_TIME_TO_TARGET_UNITS" "char")
("TORPEDO_REMARKS" "char")

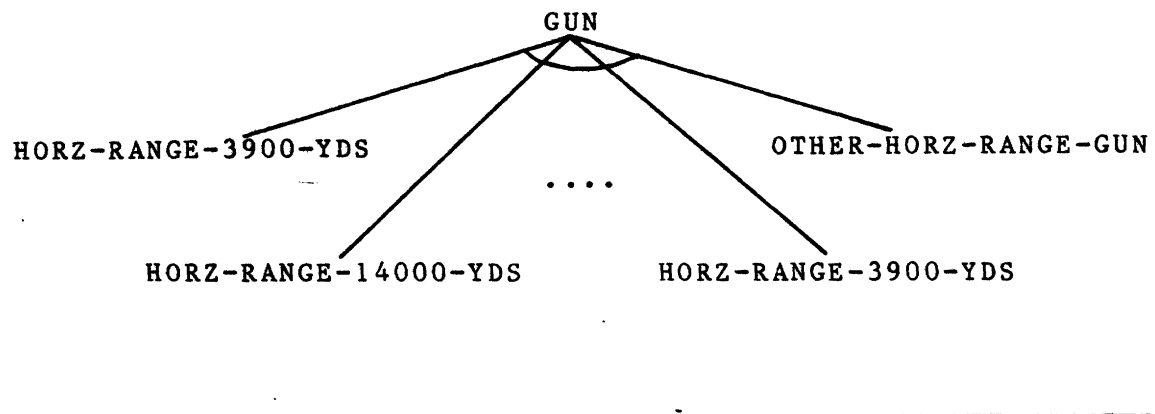
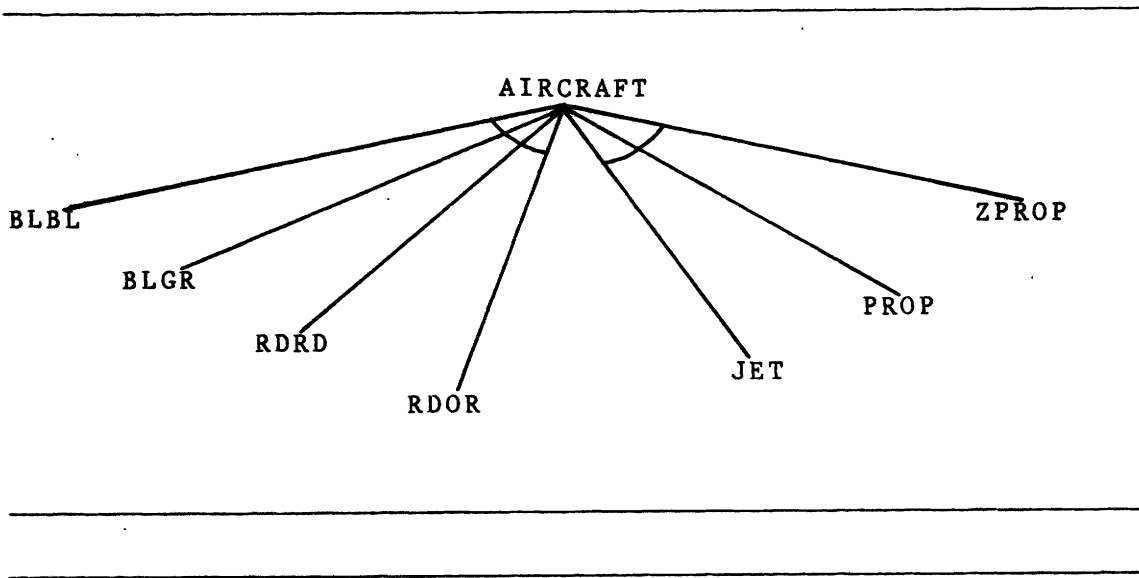
APPENDIX B

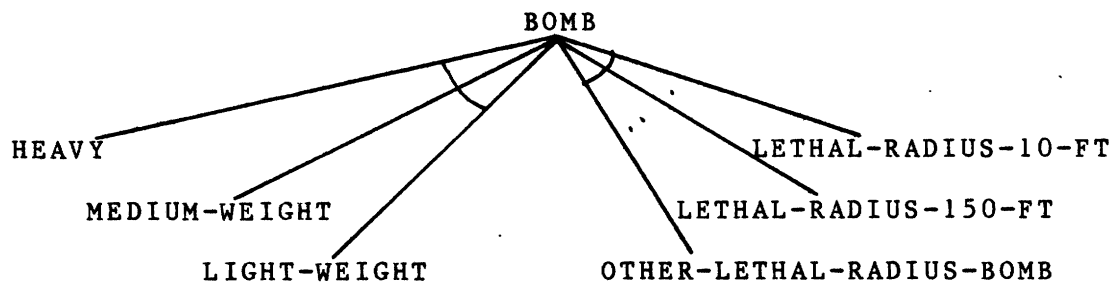
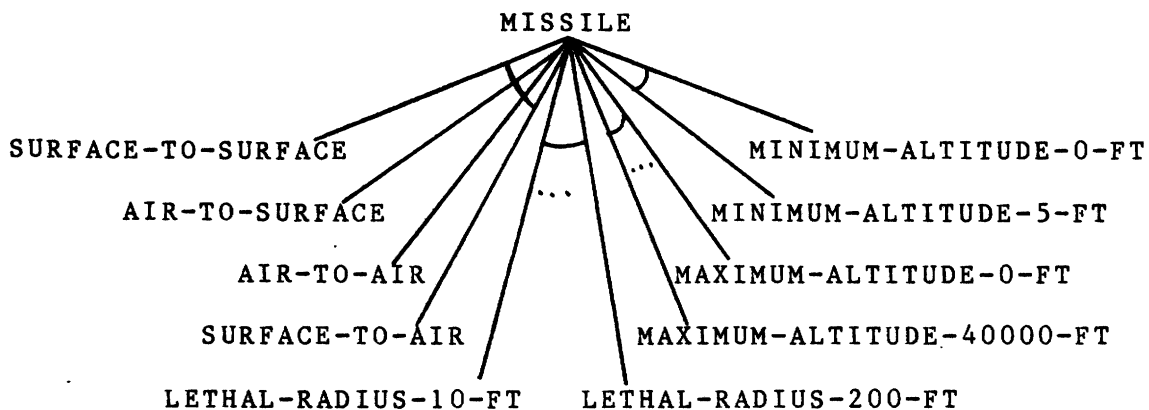
BREAKDOWNS CREATED BY ENHANCE

The following illustrate the kind of breakdowns created by ENHANCE for each entity in the database. (See Figure 4.8 for breakdowns created for entity class SHIP.) For reasons of space, all sub-classes may not be shown for each entity. In this case, dots (...) will be shown indicating a number of sub-classes are not present in the picture. In some cases the entity name has been left off of the sub-class names (also for space reasons). The ENHANCE system always includes the entity name in the name of the sub-classes formed.



BREAKDOWNS CREATED BY ENHANCE





BREAKDOWNS CREATED BY ENHANCE

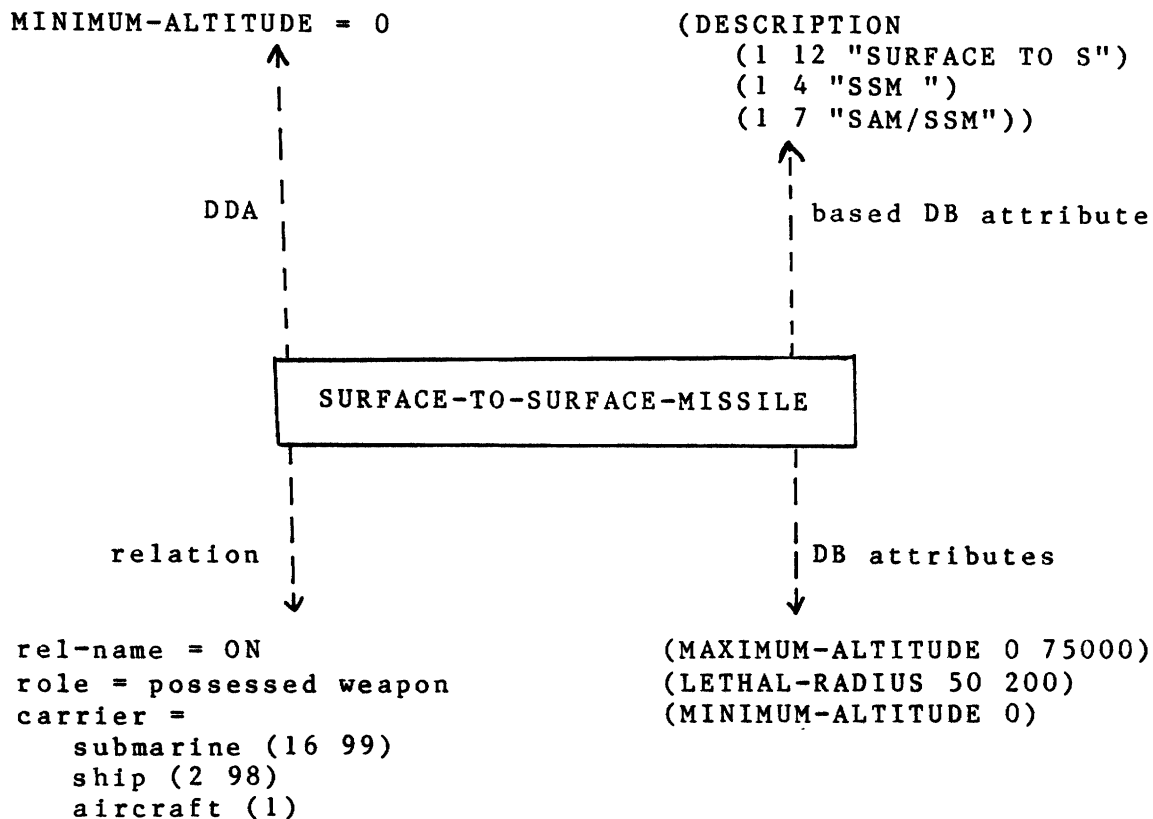


APPENDIX C

SAMPLE NODE INFORMATION CREATED BY ENHANCE

The following illustrations contain the node information associated with two of the nodes generated by ENHANCE.

SAMPLE NODE INFORMATION CREATED BY ENHANCE



SAMPLE NODE INFORMATION CREATED BY ENHANCE

