COMPUTER-AIDED RESEARCH ON SYNONYMY AND ANTONYMY *

H. P. Edmundson
University of Maryland, College Park, Md., U.S.A.

and

Martin N. Epstein
National Institutes of Health, Bethesda, Md., U.S.A.

Abstract

This research is a continuation of that reported in Axiomatic Characterization of Synonymy and Antonymy, which was presented at the 1967 International Conference on Computational Linguistics [3]. In that paper on mathematical linguistics the relations of synonymy and antonymy were regarded as ternary relations and their domains and ranges were discussed. Synonymy and antonymy were defined jointly and implicity by a system of eight axioms, which permitted the proofs of several intuitively satisfying theorems. The present paper on computational linguistics is a preliminary report which describes some computer programs that have been used to investigate the extent to which those axioms model an existing dictionary of synonyms and antonyms [9]. A set of computer programs is discussed that (1) input the dictionary data concerning synonyms and antonyms, (2) create a data structure in core memory to permit the manipulation of data, (3) query this data structure about words and relations, and (4) output the answers to queries or the entire data structure, if desired. Some examples of computer output are also given to indicate present directions of the computer-aided research.

---

*This research was supported in part by the Office of Naval Research under Contract N00014-67-A-0239-0004.

1. Introduction

1.1 Previous Research

This work is a continuation of research initially reported in the paper Mathematical Models of Synonymy, which was presented at the 1965 International Conference on Computational Linguistics [2]. That paper included a historical summary of the concepts of synonymy and antonymy. It was noted that since the first book on English synonyms, which appeared in the second half of the 18th century, dictionaries of synonyms and antonyms have varied according to the particular explicit or implicit definitions of "synonym" and "antonym" that were used. The roles of grammatical class, word context, and substitutability in the same context were discussed.

As was noted, synonymy traditionally has been regarded as a binary relation between two words. Graphs of these binary relations were drawn for several sets of words based on Webster's Dictionary of Synonyms [8] and matrices for these graphs were exhibited as an equivalent representation. These empirical results showed that the concepts of synonymy and antonymy required the use of ternary relations between two words in a specified sense rather than simply a binary relation between two words. The synonymy relation was then defined implicitly, rather than explicitly, by three axioms stating the properties of being reflexive, symmetric, and transitive. The antonymy relation was also defined by three axioms stating the properties of being irreflexive, symmetric, and antitransitive (the last term was coined for that study). It was noted that these six axioms could be expressed in the calculus of relations and that this relation algebra

could be used to produce shorter proofs of theorems, even though no proofs were given. In addition, several geometrical and topological models of synonymy and antonymy were posed and examined.

The characterizations of synonymy and antonymy initiated in Edmundson [2] were investigated more thoroughly in Edmundson [3]. Synonymy and antonymy were defined jointly and implicitly by a set of eight axioms rather than separately as before. First, it was noted that the original six axioms were insufficient to permit the proofs of certain theorems whose truth was strongly suggested by intuitive notions about synonymy and antonymy. In addition, it was discovered that certain fundamental assumptions about synonymy and antonymy must be made explicit as axioms. Some of these have to do with specifying the domain and range of the synonymy and antonymy relations. This is related to questions about whether function words, which linguistically belong to closed classes, should have synonyms and antonyms and whether content words, which linguistically belong to open classes, must have synonyms and antonyms. Several fundamental theorems of this axiom system were stated and proved. The informal interpretations of many of these theorems were intuitively satisfying. For example, it was proved that any even power of the antonymy relation is the synonymy relation, while any odd power is the antonymy relation.

These results supported the belief that an algebraic characterization is insightful and appropriate. For example, the assumption that synonymy is an equivalence relation also has been made, either directly or indirectly, by F. Kiefer and S. Abraham [4], U. Weinreich [10], and others. Since the axiom system defined the notions of syn-

onymy and antonymy jointly and implicitly, it avoided certain difficulties that are encountered when attempts are made to define these notions separately and explicitly.

1.2 Axioms

Before investigating axioms for synonymy and antonymy, we will recapitulate some notions and notations for the calculus of binary relations. Consider a set V of arbitrary elements, which will be called the universal set. A binary relation on V is defined as a set R of ordered pairs $\langle x,y \rangle$, where $x,y \in V$. The proposition that x stands in relation R to y will be denoted by xRy. The domain $\mathfrak{D}(R)$ of relation R is defined as the set $\mathfrak{D}(R) \equiv \{x:(\exists y)(xRy)\}$. The complement, union, intersection, and converse relations are defined by

$$x\bar{R}y \equiv \sim xRy$$

$$x(R \cap S)y \equiv xRy \wedge xSy$$

$$x(R \cup S)y \equiv xRy \vee xSy$$

$$xR^{-1}y \equiv yRx$$

The identity relation I is defined by

$$xIy \equiv x = y$$

The product and power relations are defined by

$$xR|Sy \equiv (\exists z)[xRz \wedge zSy]$$

$$R^n \equiv R|R^{n-1} \qquad n \geq 1$$

The inclusion and equality of relations are defined by

$$R \subseteq S \equiv xRy \Longrightarrow xSy$$

$$R = S \equiv R \subseteq S \wedge S \subseteq R$$

Under the assumption that synonymy and antonymy are ternary relations on the set of all words, the following definitions will be used:

$xS_i y \equiv$ word x is a synonym of word y with respect to the intension i (or word x is synonymous in sense i to word y)

$xA_i y \equiv$ word x is an antonym of word y with respect to the intension i (or word x is antonymous in sense i to word y)

In addition to the synonymy and antonymy relations, it will be useful to introduce the following classes that are the images by these relations. The synonym class of a word y is defined by

$$s_i(y) \equiv \{x : xS_i y\}$$

which may be extended to an arbitrary set E of words by

$$s_i(E) \equiv \{x : (\exists y)[y \in E \wedge xS_i y]\}$$

Similarly, the antonym class of a word y is defined by

$$a_i(y) \equiv \{x : xA_i y\}$$

which may be extended to a set E of words by

$$a_i(E) \equiv \{x : (\exists y)[y \in E \wedge xA_i y]\}$$

Following Edmundson [3], it will be assumed that the synonymy and antonymy relations are defined by the following set of axioms--rather than as in Edmundson [2].

Axiom 1 (Reflexive): $(\forall x)[xS_i x]$

Axiom 2 (Symmetric): $(\forall x)(\forall y)[xS_i y \Longrightarrow xS_i^{-1} y]$

Axiom 3 (Transitive): $(\forall x)(\forall y)(\forall z)[xS_i y \wedge yS_i z \Longrightarrow xS_i z]$

Axiom 4 (Irreflexive): $(\forall x)[x\bar{A}_i x]$

Axiom 5 (Symmetric): $(\forall x)(\forall y)[xA_i y \Longrightarrow xA_i^{-1} y]$

Axiom 6 (Antitransitive): $(\forall x)(\forall y)(\forall z)[xA_i y \wedge yA_i z \Longrightarrow xS_i z]$

Axiom 7 (Right-identity): $(\forall x)(\forall y)(\forall z)[xA_i y \wedge yS_i z \Longrightarrow xA_i z]$

Axiom 8 (Nonempty): $(\forall y)(\exists x)[xA_i y]$

The above eight axioms may be expressed more succinctly in the calculus of relations as follows:

Axiom 1 (Reflexive): $I \subseteq S_i$

Axiom 2 (Symmetric): $S_i \subseteq S_i^{-1}$

Axiom 3 (Transitive): $S_i^2 \subseteq S_i$

Axiom 4 (Irreflexive): $I \subseteq \bar{A}_i$

Axiom 5 (Symmetric): $A_i \subseteq A_i^{-1}$

Axiom 6 (Antitransitive): $A_i^2 \subseteq S_i$

Axiom 7 (Right-identity): $A_i | S_i \subseteq A_i$

Axiom 8 (Nonempty): $(\forall y)(\exists x)[xA_iy]$

As mentioned in [3], even though $s_i(y) \neq \emptyset$ since yS_iy by Axiom 1, it may be necessary to add the following axiom:

Axiom 9: $(\forall y)(\exists x)[x \neq y \wedge xS_iy]$

to guarantee that the domain of the relation S_i is not trivial, i.e.,

$$s_i(y) - \{y\} \neq \emptyset$$

Axiom 9 is not necessary if $s_i(y)$ is permitted to be a unit set for certain words. Thus, we might define $s_i(y) = \{y\}$ for any function word y, e.g., $s_i(\text{and}) = \{\text{and}\}$. But this will not work for antonymy since $a_i(y)$ might be considered empty for certain words such as function words, e.g., $a_i(\text{and}) = \emptyset$. The alternative of defining $a_i(y) = \overline{\{y\}}$ is not reasonable since it produces more problems than it solves.

Axiom 8: $(\forall y)(\exists x)[xA_iy]$, which is equivalent to

$$(\forall y)[a_i(y) \neq \emptyset]$$

is reasonable if the contrary $\bar{y}$ of word y (e.g., "irrelevant", "impossible", "nonuse", etc.) is permitted, i.e., $\bar{y} \in a_i(y)$.

2. Research Methodology

2.1 Research Goals

The synonymy and antonymy relations possess interesting properties, which can be treated mathematically to provide insight about semantic relations and connectivity among words in a natural language. One such model is the axiom system just stated. The immediate goal of the current research is to compile, in computer-accessible form, a dictionary containing all synonymy and antonymy relations holding between selected words. Such a dictionary is useful in gaining a better understanding of how the English lexicon is semantically structured since it can eventually enable the determination of the completeness of the descriptions in any synonym-antonym dictionary. Another objective is to assist the lexicographer in compiling such a dictionary so that all words are defined and related in a consistent manner.

2.2 Data Base and Data Structure

For the present research a test dictionary was compiled by selecting English words from Webster's New Dictionary of Synonyms [9]. Accordingly, a set of computer programs was written to do the following:

1. Input, in a prescribed format, words selected from the above dictionary together with relevant data concerning their synonyms and antonyms.

2. Create in core memory a suitable data structure (see [5]) for the input, which permits the manipulation of the dictionary data. Future extensions to the system would make use of direct-access storage to enable the processing of more data.

2.3 Data Analysis

The test dictionary is analyzed with the aid of computer programs that were written to do the following:

1. Query the data structure about words and relations. Two query modes are built into the system. The first mode allows the selection of words fulfilling an input request and the second mode permits the verification that certain relations hold between selected words.

2. Output the answers to queries or output the entire data structure, if desired.

3. Verify the consistency of word groupings, the degree of completeness of related subgroups, and the presence or absence of anomalies in the data base.

3. Input

3.1 Input Specification

First, it is necessary to specify and format the input data so that a set of programs may process and query a test dictionary, which resides in core in the present version of the system. This is accomplished using the following input prototype:

<word>,<grammar code><sense #><relation>,<word>,...,<word>,;

where

1. <word> is an entry in Webster's New Dictionary of Synonyms.

2. <grammar code> makes use of the following coding mnemonics:

N - Noun
V - Verb
J - Adjective
B - Adverb
O - Pronoun

D - Determiner
L - Auxillary
P - Preposition
C - Conjunction

3. <sense #> is a one-digit number representing a sense associated with a word in the dictionary.

4. <relation> is denoted by

S - Synonymy
A - Antonymy
M - word used in the description of another word but not itself a main entry.

5. <word>,...,<word> is the set of words standing in the given relation to the main entry in the given sense.

Thus, each input item consists of a main-entry word followed by a comma, a one-character grammar code, a one-digit sense number, a one-character relation, a comma, a list of words (separated by commas) that in the given sense stand in the given relation to the main entry, a comma, and a semicolon that denotes the end of an input item. A sample computer input is:

SIMPLE,J2S,EASY,FACILE,LIGHT,EFFORTLESS,SMOOTH,;

Continuation cards may be appended to any item by placing a "+" in column 80 of subsequent cards.

3.2 Comments

Several problems remain in fully attaining the above stated goals. On the one hand, it is difficult to select from a manual dictionary sufficiently small sets of words that are closed under the relations S and A, while on the other hand large segments of such a dictionary cannot be input at present. Programs have been written to structure and process small test dictionaries, to select words from the data

structure using a query language, and to verify that certain relations hold between words.

4. Processing

4.1 Input Analysis

In the first phase of processing the program checks the well-formedness of the input entries, isolates words, records grammatical classes, and establishes relations between words.

4.2 Creation of the Data Structure

The data structure created in core provides for the construction of two tables.

The first is a directory table whose items consist of a location identifier, an entry, the grammar code, the sense number, and the relation. This directory sequentially stores the input information, eliminates duplicates, and provides a reference pointer to a second table, the matrix table.

The matrix table consists of an incidence or connectivity matrix, which is used to store the synonymy and antonymy relations between words. It should be noted that xSy is stored differently from ySx. In addition xSx is recorded in the data structure only if it so appeared in Webster's New Dictionary of Synonyms.

It is also possible to develop a reachability or accessibility matrix from the stored input. In graph-theoretic terms the matrix may be regarded as follows: words correspond to vertices and relations correspond to directed edges. Note that for all x and y in the data structure, it can be determined whether xSy and xAy are true or false.

4.3 Query Language

The two basic modes of operating upon the data structure are the selection mode and the verification mode. Both modes permit queries to be composed and matched against the data structure. The response to a query statement in the selection mode is a listing of all those entries in the data structure satisfying the request. In the verification mode the response indicates whether a statement is true or false and, if false, points out which segment of the query statement does not hold.

Simple query statements are of the form:

? if QUERY

where "?" is used to initiate the request; "if" is used as a prefix for particular query types; and "QUERY" consists, in the simplest case, of one of the following five statement types:

1. xRy
2. x*y
3. xR*
4. *Ry
5. *R*

where "*" denotes that any value in the specified field is allowed and the sense i is not explicitly denoted. Item 1 above operates in the verification mode, while items 2-5 operate in the selection mode.

Simple query statements can be extended to allow compound expresions by means of the operators "not", "and", and "then". For example, the query

? if xSy and ySx

tests whether synonymy (in sense i) is a symmetric relation for the words x and y, while the query

? if xSy and ySz then xSz

checks to see if transitivity of synonymy (in sense i) holds for the words x,y, and z.

It is also possible to determine if the composition S|S of the relation S holds, i.e., for given words x and y, does the given word z in the data structure satisfy the request:

? if xSz and zSy

To select all such z from the data structure, the request is formulated as follows:

? if xS* and *Sy

The synonymy relation S is assumed to be reflexive, symmetric, and transitive, while the antonymy relation A is assumed to be irreflexive, symmetric, and antitransitive. The input forms of queries representing these properties are as follows:

1. Reflexive: ? x,S,x,;
2. Symmetric: ? if x,S,y,.then.y,S,x,;
3. Transitive: ? if x,S,y,.and.y,S,z,.then.x,S,z,;
4. Irreflexive: ? not x,A,x,;
5. Symmetric: ? if x,A,y,.then.y,A,x,;
6. Antitransitive: ? if x,A,y,.and.y,A,z,.then.x,S,z,;

In addition, the input format for the properties of right-identity and nonempty are as follows:

7. Right-identity: ? if x,A,y,.and.y,S,z,.then.x,A,z,;
8. Nonempty: ? if *,A,y,;

This last property is interpreted as follows: for all y in the data

structure, does there exist a word x such that x stands in the relation A to y?

An example of the input to test if transitivity holds for the words "big", "great", and "large", in that order is as follows:

? if big,S,great,.and.great,S,large,.then.big,S,large,;

4.4 Verification Algorithms

Two basic verification algorithms have been programmed. The first seeks to detect the presence of either a chain or a loop among the given words. The input consists of pairs of words standing in the relation S. A chain exists if it is possible to linearly order the set of input words so that the relation S holds between adjacent words. A loop is detected if every word is preceded by another word and the algorithm cannot locate a word that has no predecessor. This algorithm may be useful in developing techniques for structuring the vocabulary of a synonym-antonym dictionary so that no word is used before it has been defined.

The second algorithm determines whether selected groups of words form an equivalence class with respect to synonymy in a given sense. A binary relation R is said to be an equivalence relation if it is reflexive, symmetric, and transitive. An equivalence relation R partitions a set of elements into disjoint classes such that two elements are equivalent if and only if they belong to the same class. The routine determines whether two given words are in an existing synonym class and, if not, establishes a new class. The test for equivalence classes in a set of words is initiated by the input statement.

EQUV(<word>,...,<word>)

which incorporates tests for reflexivity, symmetry, and transitivity. The output is a table indicating class membership of words or, if no equivalence relations exist, indicates those properties not satisfied by particular words. For example, the routine found that, aside from reflexivity, the words "pure", "simple", and "absolute" formed an equivalence class in a particular sense i. On the other hand, the words "aft", "astern", "abaft", "after", and "behind" formed two equivalence classes {aft, astern, abaft} and {after, behind}. At present, the graphs of equivalence classes are drawn manually, rather than by computer.

Appendix 2 outlines the structure of an input deck and lists a sample input including both input data and query statements.

5. Output

5.1 Relational Form

The relational form of output verifies whether the simple statement xRy is true or false and also whether compounds of simple statements are true or false. For example, the query

```
? if stern A soft
```

produced

```
THE FOLLOWING RELATION HOLDS:   STERN A SOFT
```

while for the query

```
? if far A high and high A low then high A far
```

the following set of responses was obtained:

```
THE FOLLOWING RELATION HOLDS:   FAR A HIGH
THE FOLLOWING RELATION HOLDS:   HIGH A LOW
THE GIVEN RELATION DOES NOT HOLD:   HIGH A FAR
QUERY REQUEST NOT SATISFIED -- STATEMENT FALSE
```

5.2 List Form

In the list form of response to a query the main entry and all words (if any) that are pointed to by the main entry are listed. For example, the query

? if * S stern

produced

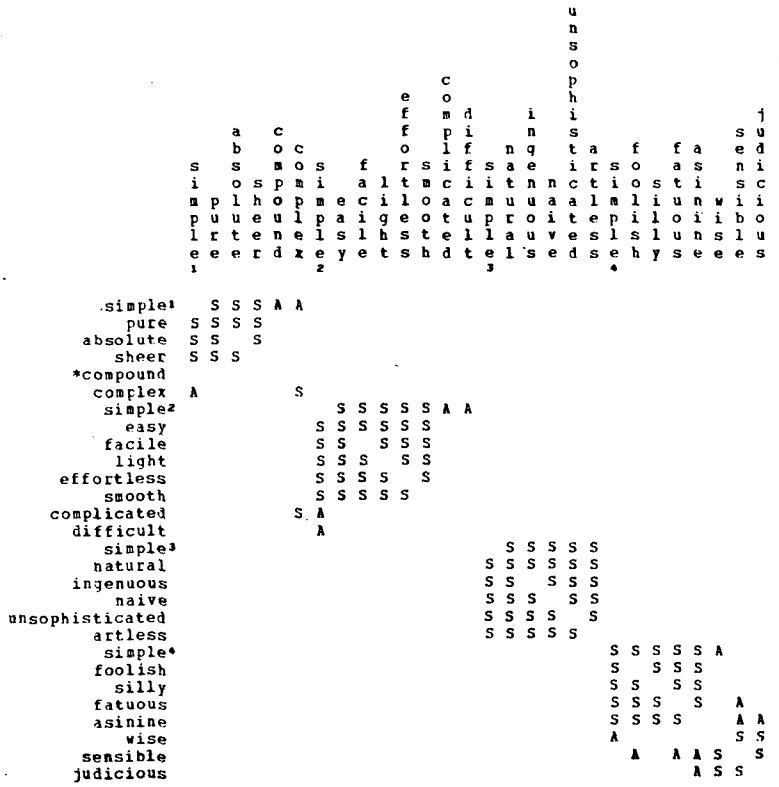
THE FOLLOWING WORDS ARE IN THE RELATION S TO STERN

SEVERE
AUSTERE
ASCETIC

In general, this form of output consists of lists of the following two types: a list of all words synonymous or antonymous to a given word, and a list of all synonymy or antonymy relations holding among a given set of words.

5.3 Matrix Form

The matrix form of output represents the relations by a matrix consisting of S's and A's according to whether the relation S or A holds between given pairs of words. A blank in such a matrix indicates that neither S nor A relates two words in the data structure. For example, the following matrix revealed four senses of the word "simple".



The superscript denotes the sense number to be associated with "simple". A "*" is placed to the left of those words that do not appear as main entries in Webster's New Dictionary of Synonyms.

6. Concluding Remarks

The programs were written almost completely in FORTRAN IV and have been run on the IBM 360 and the PDP 10. A flowchart, which summarizes these programs, appears as Appendix 1. In addition, a SNOBOL 4 program has been written for the detection of chains and loops.

Several problems in fully achieving the stated research goals have appeared. It was difficult to select small closed sets of words from Webster's New Dictionary of Synonyms and it was not feasible to keypunch the entire dictionary. Since the size of a truly suitable data base was too large to retain in core memory, several sample dictionaries have been selected to study the feasibility of the principles and techniques involved. Most of the current effort has been devoted to providing programming capability for the processing of small test dictionaries. Different words may be input with each run, thereby increasing the size of the sample data base to gain deeper insight into the properties of the entries listed in a manual dictionary. Further computer-aided research on synonyms and antonyms will help to validate or extend the axiomatic model proposed earlier. Also, future research could consider the additional relations "contrasting" and "analogous" cited in some manual dictionaries and the automatic determination of the senses of words.

Bibliography

[1]. R. Carnap, Introduction to Symbolic Logic and Its Applications, W. Meyer and J. Wilkinson (trs.). Dover, N.Y., 1958.

[2]. H. P. Edmundson, "Mathematical Models of Synonymy", International Conference on Computational Linguistics, New York, 1965.

[3]. H. P. Edmundson, "Axiomatic Characterization of Synonymy and Antonymy", International Conference on Computational Linguistics, Grenoble, 1967.

[4]. F. Kiefer and S. Abraham, "Some Problems of Formalization in Linguistics", Linguistics, v. 17, Oct. 1965, pp. 11-20.

[5]. D. Knuth, The Art of Computer Programming : Vol. 1, Fundamental Algorithms, Addison-Wesley, New York, 1968.

[6]. V. V. Martynov, Pytannja prikladnoji lingvistyky; tezisy dopovidej mižvuzovs'koji naukovoji konferenciji, Sept. 22-28, 1960, Černivcy.

[7]. A. Naess, "Synonymity as Revealed by Intuition", Philosophical Review, v. 66, 1957, pp. 87-93.

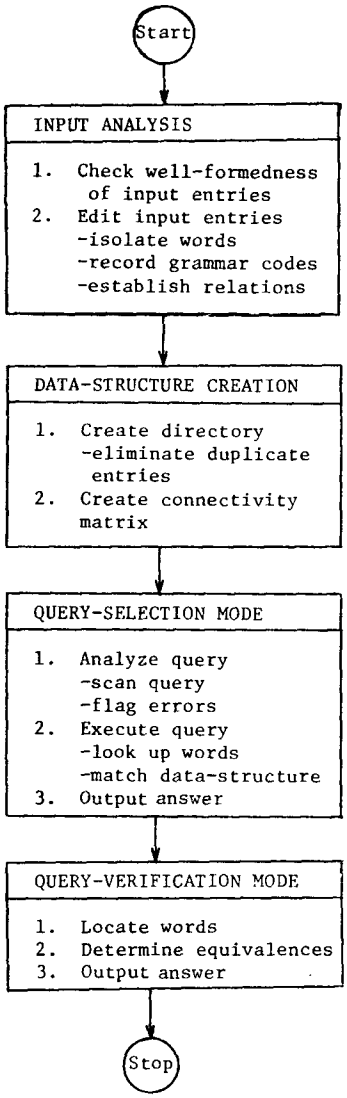
[8]. Webster's Dictionary of Synonyms, Merriam Co., Springfield, Mass., 1951.

[9]. Webster's New Dictionary of Synonyms, Merriam Co., Springfield, Mass., 1968.

[10]. U. Weinreich, "Explorations in Semantic Theory", in Current Trends in Linguistics, III, T. Sebeok (ed.), Mouton and Co., The Hague, 1966.

[11]. P. Ziff, Semantic Analysis, Cornell University Press, Ithaca, N.Y., 1960.

APPENDIX 1-FLOW CHART



APPENDIX 2 - RUN INSTRUCTIONS

First the basic structure of an input deck is outlined. Comments and explanation are enclosed in parentheses. A sample run which may be input to the system follows.

---------- FORMAT OF THE INPUT DECK ----------
PARAMETER CARD (constant for each run) .
(input to the creation program in the input prototype described above)

. . .
. . .
. . .

EOD (input delimeter) .
-SELECT MODE
(query requests in both the verify and select mode follow)
(all queries which are to be matched against the data file. using the query input format described above.)
-EQUIVALENCE MODE
(using the set of terms inputted, determine if the set forms an equivalence class)
** (run terminator) .

A SAMPLE RUN

```
.; +ZSAEONIF*.
FOR,C S,BECAUSE,SINCE,AS,INASMUCH AS,;
BECAUSE,  S,FOR,SINCE,AS,INASMUCH AS,;
SINCE,  S,BECAUSE,FOR,AS,INASMUCH AS,;
AS,  S,SINCE,BECAUSE,FOR,INASMUCH AS,;
INASMUCH AS,  S,SINCE,BECAUSE,FOR,AS,;
FULL,  S,COMPLETE,PLENARY,REPLETE,;
FULL,  A,EMPTY,;
COMPLETE,  S,FULL,PLENARY,REPLETE,;
COMPLETE,  A,INCOMPLETE,;
REPLETE,  S,FULL,COMPLETE,PLENARY,;
PLENARY,  S,FULL,COMPLETE,REPLETE,;
PLENARY,  A,LIMITED,;
EMPTY,  S,VACANT,BLANK,VOID,VACUOUS,;
EMPTY,  S,EMPTY,;
INCOMPLETE,  M,;
LIMITED,  M,;
SEVERE,  S,STERN,AUSTERE,ASCETIC,;
SEVERE,  A,TOLERANT,;
STERN,  S,SEVERE,AUSTERE,ASCETIC,;
STERN,  A,SOFT,;
AUSTERE,  S,SEVERE,STERN,ASCETIC,;
AUSTERE,  A,LUSCIOUS,;
ASCETIC,  S,AUSTERE,SEVERE,STERN,;
```

```
ASCETIC,  A,LUXURIOUS,;
REMOTE,  S,DISTANT,FAR,REMOVED,;
REMOVED,  S,REMOTE,FAR,DISTANT,;
DISTANT,  S,FAR,REMOVED,REMOTE,;
FAR,  S,REMOTE,REMOVED,DISTANT,;
FAR,  A,NEAR,HIGH,NEARBY,;
NEAR,  S,CLOSE,HIGH,NEARBY,;
NEAR,  A,FAR,;
HIGH,  S,TALL,LOFTY,;
TALL,  S,HIGH,LOFTY,;
NEARBY,  S,CLOSE,NEAR,HIGH,;
HIGH,  A,LOW,;
CLOSE,  S,NEAR,HIGH,NEARBY,;
CLOSE,  S,CLOSE,;
CLOSE,  A,REMOTE,;
SIMPLE,J1S,PURE,ABSOLUTE,SHEER,;
SIMPLE,J1A,COMPOUND,COMPLEX,;
PURE,J S,SIMPLE,PURE,ABSOLUTE,SHEER,;
ABSOLUTE,J S,SIMPLE,PURE,SHEER,;
SHEER,J S,SIMPLE,PURE,ABSOLUTE,;
COMPLEX,J S,COMPLEX,COMPLICATED,INTRICATE,INVOLVED,KNOTTY,;
COMPLEX,J A,SIMPLE,;
COMPOUND,  M,;
Z
SIMPLE,J2S,EASY,FACILE,LIGHT,EFFORTLESS,SMOOTH,;
SIMPLE,J A,COMPLICATED,DIFFICULT,;
COMPLICATED,  S,INTRICATE,INVOLVED,COMPLEX,KNOTTY,;
COMPLICATED,  A,SIMPLE,;
COMPLEX,  S,COMPLEX,COMPLICATED,INTRICATE,INVOLVED,KNOTTY,;
FACILE,  S,EASY,SMOOTH,LIGHT,SIMPLE,EFFORTLESS,;
LIGHT,  S,EASY,SIMPLE,FACILE,EFFORTLESS,SMOOTH,;
EFFORTLESS,  S,EASY,SMOOTH,FACILE,SIMPLE,LIGHT,;
SMOOTH,  S,EFFORTLESS,EASY,LIGHT,SIMPLE,FACILE,;
Z
SIMPLE, 3S,NATURAL,INGENUOUS,NAIVE,UNSOPHISTICATED,ARTLESS,;
NATURAL,  S,NATURAL,SIMPLE,INGENUOUS,NAIVE,UNSOPHISTICATED,ARTLESS,UNAFFECTED,;
INGENUOUS,  S,NATURAL,SIMPLE,NAIVE,UNSOPHISTICATED,ARTLESS,;
NAIVE,  S,UNSOPHISTICATED,ARTLESS,INGENUOUS,NATURAL,SIMPLE,;
UNSOPHISTICATED,  S,NATURAL,SIMPLE,INGENUOUS,NAIVE,ARTLESS,;
ARTLESS,  S,NATURAL,SIMPLE,INGENUOUS,NAIVE,UNSOPHISTICATED,UNAFFECTED,;
UNAFFECTED,  S,ARTLESS,NATURAL,SIMPLE,INGENUOUS,NAIVE,UNSOPHISTICATED,;
Z
SIMPLE,J4S,FOOLISH,SIMPLE,SILLY,FATUOUS,ASININE,;
FOOLISH,  S,SIMPLE,SILLY,FATUOUS,ASININE,;
SILLY,  S,SIMPLE,FOOLISH,FATUOUS,ASININE,;
FATUOUS,  S,ASININE,SILLY,FOOLISH,SIMPLE,;
FATUOUS,  A,SENSIBLE,;
ASININE,  S,SIMPLE,FATUOUS,SILLY,FOOLISH,;
ASININE,  A,SENSIBLE,JUDICIOUS,;
SIMPLE, 4A,WISE,;
WISE,  S,SENSIBLE,JUDICIOUS,;
WISE,  A,SIMPLE,;
SENSIBLE,  A,FOOLISH,FATUOUS,ASININE,;
SENSIBLE,  S,WISE,JUDICIOUS,;
JUDICIOUS,  S,WISE,SENSIBLE,;
```

```
JUDICIOUS,  A,ASININE,;
INFERIOR,  S,UNDERLING,SUBORDINATE,;
INFERIOR,  S,INFERIOR,;
INFERIOR,  A,SUPERIOR,;
UNDERLING,  S,INFERIOR,SUBORDINATE,;
UNDERLING,  A,LEADER,MASTER,;
SUBORDINATE,N1S,INFERIOR,UNDERLING,;
SUBORDINATE,J1S,SECONDARY,DEPENDENT,SUBJECT,TRIBUTARY,COLLATERAL,;
SUBORDINATE,J1S,SUBORDINATE,;
SUBORDINATE,J1A,CHIEF,LEADING,;
SUBORDINATE,J2A,DOMINANT,;
SUPERIOR,  S,BETTER,PREFERABLE,;
SUPERIOR,  A,INFERIOR,;
PREFERABLE,  S,BETTER,SUPERIOR,;
BETTER,  S,SUPERIOR,PREFERABLE,BETTER,;
LEADER,  S,HEAD,CHIEF,CHIEFTAIN,MASTER,;
LEADER,  A,FOLLOWER,;
MASTER,  S,CHIEF,CHIEFTAIN,HEAD,LEADER,;
FOLLOWER,  S,ADHERENT,DISCIPLE,SECTARY,PARTISAN,HENCHMAN,SATELLITE,;
FOLLOWER,  A,LEADER,;
EOD

-SELECT MODE FOR QUERY
? FAR,S,REMOTE,;
? STERN,*,SEVERE,;
? STERN,A,SOFT,;
? *,S,STERN,;
? STERN,S,*,;
? *,S,*,;
? STERN,S,AUSTERE,;
? FULL,A,EMPTY,;
? IF FAR,A,NEAR,.AND.CLOSE,A,REMOTE,.THEN.FAR,S,REMOTE,;
? IF FAR,A,NEAR,.THEN.NEAR,A,FAR,;
? FAR,A,HIGH,;
? FAR,A,*,.AND.*,S,HIGH,;
? IF FAR,A,HIGH,.AND.HIGH,A,LOW,.THEN.HIGH,A,FAR,;
? IF UNDERLING,A,LEADER,.AND.LEADER,A,FOLLOWER,.THEN.UNDERLING,S,FOLLOWER,;
? SIMPLE        J1,S,SIMPLE        J1,;
? SIMPLE        J4,S,SIMPLE        J4,;
? IF SIMPLE        J1,A,COMPLEX,.AND.COMPLEX,A,SIMPLE        J4,.THEN.
        SIMPLE        J1,S,SIMPLE        J1,;
? IF ASININE,A,SENSIBLE,.AND.SENSIBLE,S,JUDICIOUS,.THEN.ASININE,A,JUDICIOUS,;

-EQUIVALENCE MODE
EQUV(FULL,COMPLETE,FULL,PLENARY,FULL,REPLETE,COMPLETE,PLENARY,REPLETE,FULL,)
EQUV(HIGH,TALL,HIGH,LOFTY,LEADER,HEAD,LEADER,MASTER,)
**
```